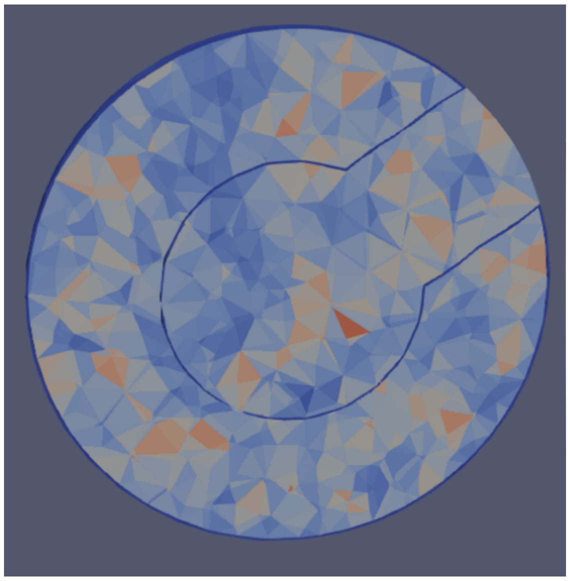
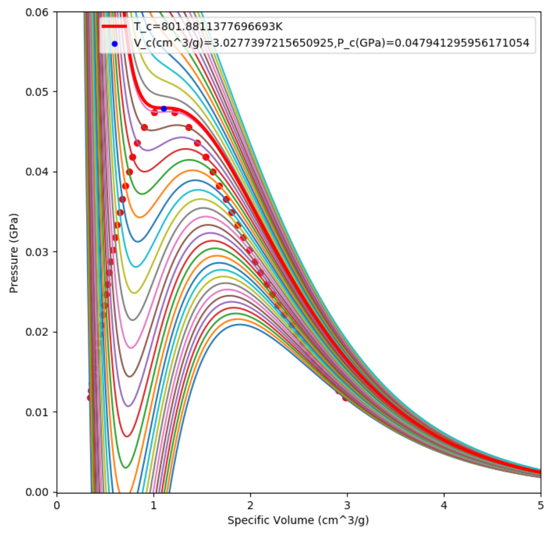
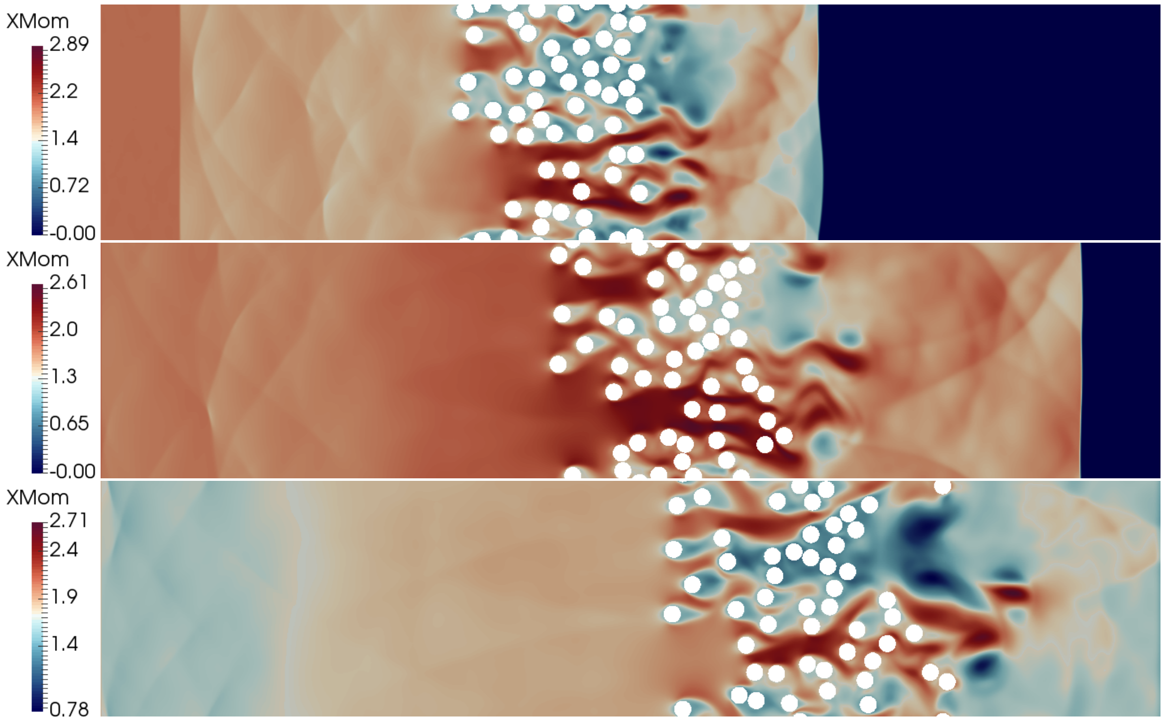
Title:	Final Reports of the 2020 Los Alamos National Laboratory Computational Physics Student Summer Workshop
Author(s):	Abrams, Joshua Rubin; Bell, Brian Wesley; Blade, Tracy Shane; Dyer, Joshua Wade; Holguin, Francisco; Keithley, Jeffrey Scott; Leiendecker, Harrison; Johns, William Richard; Koskelo, Eliseanne Corinne; Lo, Nga Ying; Martin, Olivia Grace; May, Asher Paul; Natan, Tali Marshall; Nguyen, Le Viet; Ottoway, Crystal Faith; Owens, Dylan Aubrey; Ozier, Samuel Douglas; Somers, Alex Daniel; Taylor, Timothy Joseph; Warhover, Alex Jacob; White, Jackson Richard; et al.
Intended for:	Report
Issued:	2020-10-20





Final Reports of the 2020 Los Alamos National Laboratory Computational Physics Student Summer Workshop



Workshop Coordinators: Daniel Israel, Madison Andrews, and Garry Maskaly

Contents

Preface

The Los Alamos National Laboratory (LANL) Computational Physics Workshop has been hosted by the X-Computational Physics Division (XCP) since 2011. That makes this the tenth year of the workshop. Any potential for a special celebration to mark the event was short lived; in late March, in response to the COVID-19 pandemic, LANL announced a pause in all student programs.

Working closely with management, we were able to transition the workshop to an entirely off-site, virtual format for the summer of 2020. This entailed many changes, not just in the student experience, but also in the lecture schedule and mentoring, as well as requiring the replacement or redesign of several projects. Our administrative and computer support staff worked diligently to make the switch to a remote work posture for the students. And our students and mentors went above and beyond to make the program a success.

While we look forward to returning to more normal, on-site, program very soon, we can unhesitatingly say that this year's workshop was a wonderful experience for everyone involved, and a terrific demonstration of LANL's resiliency in the face of this pandemic. The workshop organizers would like to thank everyone, students, mentors, and both technical and support staff, for making the 2020 XCP Computational Physics Workshop a reality.

Daniel M. Israel
Madison Andrews
Garry Maskaly
Workshop Coordinators

About the Workshop

For the past ten years, the workshop has been bringing a highly talented and diverse group of student every summer. Students work in teams of two, alongside typically two mentors, on research projects reflecting a broad range of topics within computational physics. In addition, students attend a series of lectures on topics within computational physics, facility tours, and networking events. The program lasts ten weeks, with this year's workshop running from June 8 to August 14. At the end of the summer, students give a final presentation, along with a written report. Those reports are what make up the remaining sections of this document.

Admission to the workshop is by a competitive process, with the mentors forming the selection committee. One of the important accomplishments of the workshop has been to create a student pipeline from diverse schools that sometimes are not normally tapped by LANL recruiting. Many workshop students maintain a continuing relationship with LANL, returning as students interns, post-doctoral researchers, and staff members. Additionally, workshop alumni act as ambassadors for LANL. The result is a wider awareness both of LANL as a potential employer, and of the technical work that happens at LANL.

This year, the workshop format was changed in several ways, in order to accommodate the off-site, virtual format. Students worked on LANL virtual desktop systems remotely, also accessing LANL HPC resources. In order to facilitate communication, student were given accounts on both Webex, a video teleconferencing platform, and Mattermost, an online team collaboration and chat platform, similar to Slack. Daily communication between students and mentors was primarily on Mattermost, with Webex conferencing as needed.

The lectures (Table 1) were all done on Webex. Given the difficulty of the virtual format, and a concern that students might have video teleconferencing burn-out after an academic3 semester largely moved to that format, all lectures were optional this year. In spite of this, the attendance was generally high. Lecturers were asked to try to move to a more high-level, ”What is it?,” format.

Once again, the students did a tremendous job. Over the course of ten weeks, they did important research across a staggering array of disciplines. The following pages contain the final report for each team's research efforts. We hope you will find reading them as exciting as it was for us to produce them.

Title	**Lecturer**
Computational Physics at LANL	Scott Doebling (XCP-DO)
High-Performance Computing at LANL	Joel Kulesza (XCP-3)
Software Carpentry	Chris Malone (XCP-1), Steven Andrews (XCP-8), & Pat Grubel (CCS-7)
Applied Machine Learning in Keras	Garry Maskaly (XCP-8)
Introduction to Numerical Methods	Daniel Israel (XCP-4)
Monte Carlo Methods	James Hill (XCP-1)
Computational Molecular Physics	Jeffery Leiding (T-1)
History of MCNP	Avneet Sood (XCP-7)
Lagrangian Hydrodynamics	Nathaniel Morgan (XCP-4)
Molecular Dynamics	Saryu Fensin (MST-8)
Modeling Material Strength & Damage	Abby Hunter (XCP-5)
Simulating an ICF Capsule	Brian Haines (XCP-2)
Materials Modeling and Equations of State	Kevin Honell (XCP-5)
Nuclear Data Evaluation	Denise Neudecker (XCP-5)
Proton Radiography Experiments for Understanding Dynamic Material Behavior	Katherine Prestridge (P-23)
Nuclear Safeguards	Alexis Trahan (NEN-1)
Modeling Weapons Effects	Jon Reisner (XCP-4)
Opacitities	Chris Fontes (XCP-5)
Science at LANL	John Sarrao, Deputy Lab Director for Science, Technology, & Engineering
Uncertainty Quantification	Kendra van Buren (XCP-8)
Verification and Validation	Nathan Woods (XCP-8)
Turbulence	Daniel Israel (XCP-4)
Supernova	Chris Fryer (CCS-2)
Science-Based Stockpile Stewardship	Robert Webster, Deputy Lab Director for Weapons
Experimental Facilities at LANL	Brian Jensen (M-9)
Shock Waves	Len Margolin (XCP-5)
Medical Imaging Machine Learning	Garry Maskalay (XCP-8)
Machine Learning Interpretability	Kyle Hickman (XCP-8)
History of LANL and Y-site Spies	Alan Carr, Laboratory Historian
Climate Change and Ice Modeling	Elizabeth Hunke (T-3)
Parallel Computing	Robert Robey (XCP-2)
COVID Modeling	Carrie Manore (A-1)
Neutron Diagnosed Subcritical Experiments	Madison Andrews (XCP-7)

Table 1: 2020 Lecture Series

Chapter 1

Why do explosions look like earthquakes?

Team Members
J. Rubin Abrams and Asher May

Mentors
Carène Larmat and Ting Chen

Abstract

In this study, seismic wave propagation is simulated at the DAG Site within the Nevada National Security Site using the open-source software, SPECFEM3D, which implements an optimized Spectral Element Method with MPI to run large simulations at low computational cost. We test the influence of the geologic setting on discriminating between explosions and earthquakes by simulating explosions of constant depth and intensity and varying the elastic properties of the subsurface. We also simulate explosions of varying depth and intensity for constant elastic properties of the geologic setting. We use the common metric P/S to study the ratios of P- and S-wave amplitudes generated from explosions and earthquakes and determine for what distances one can distinguish between the two.

We find that sharper contrasts in the elastic properties between geologic layers generates more S waves. This results in poorer earthquake-explosion discrimination at large distances from the borehole. We also find that deeper explosions force wave fields to interact more with geologic layers, generating more S waves in the process. Finally, we observe that even minute variations in similar seismic events such as earthquakes cause drastically different P/S signals. This is attributed to the inherent complexity of wave propagation in a non-homogeneous 3-D medium.

1.1 Introduction

During the mid 1950s, a nuclear arms race between the US, England, and the Soviet Union was reaching dangerous levels, a threat of breakout of nuclear war. In 1963, the Limited Test Ban Treaty was signed to prohibit further nuclear testing in outer space, the atmosphere, or underwater [Editors (2009)]. Later, in 1996, the United Nations signed the Comprehensive Nuclear-Test-Ban Treaty to prohibit any form of nuclear weapon testing [Wikipedia.com (2020)]. Since then, seismological observations have played an important role in monitoring nuclear explosions. In an effort to improve explosion monitoring, the US Department Of Energy's National Nuclear Security Administration (NNSA) conducted Source Physics Experiment (SPE) consisting of chemical explosions occurring at various depths and intensities. The aim of these experiments was to create numerical algorithms reflecting the true nature of the physics of the source while taking into account the surrounding geological medium. Ultimately, these numerical techniques would simulate explosions at any depth, intensity, and in any geologic setting [Snelson et al. (2013)].

In this paper, we attempt to answer the question: "Why do explosions look like earthquakes?". We create simulations of earthquakes and explosions at the NNSA's Dry Alluvium Geology (DAG) Site using the open-source software, SPECFEM3D, to generate seismic velocity wave fields. Then, we analyze them to quantify the similarities and differences in the sources. Previously, seismic signal analysis has successfully discriminated between man-made and seismic events. The most common technique involves analyzing the ratio between the primary (P), compression, and secondary (S), shear, amplitudes of seismic waves. We use the metric $\log(P/S)$ on our seismograms to classify the sources.

In this study, explosions and earthquakes are simulated using the open-source software, SPECFEM3D, which implements an optimized Spectral Element Method with MPI to run large simulations at low computational cost [Komatitsch and Vilotte (1998)]. This software will help us understand the interaction of the seismic wave field with the surrounding geologic setting. The software performs full waveform modeling of the propagation of seismic waves in the complex 3D medium. We will use this to test the influence of the 3D structure of the subsurface on discrimination between earthquakes and explosions.

In previous work, Houng demonstrates a periodic property in the Fourier Transform of the P/S ratios that occur in earthquakes and not in explosions [Houng (2018)]. He exploits this by creating a relative errors metric to classify how close a seismograph is to having this property. Houng used seismic recordings from stations spread across the Korean Peninsula to classify the sources of seismic waves. Houng's technique classified 80% of explosions and 90% of natural earthquakes correctly. Houng also determined best sampling frequency and minimum azimuthal coverage needed for accurate classification of the source.

In another work by O'Rourke et al. [O'Rourke et al. (2016)], the P/S ratios of seismograph recordings spread across the Big Horn Mountains, WY were used to distinguish between borehole shots, mining blasts, and natural earthquakes. The authors were able to determine factors that influence accurate classification such as best frequency filtering for separation of data, minimum number of stations needed for classification, and minimum azimuthal coverage necessary for classification. Interpretation of these results had to be conducted in the context of the surrounding geologic environment. The study found the amplitudes of waves from borehole shots and mining blasts to be much weaker for recording stations on the other side of the mountain range.

In a previous work, SPECFEM3D was used to simulate seismic activity throughout the entire Earth [Komatitsch et al. (2003)]. A 3D mesh structure consisting of 5.5 billion mesh points (14.6 billion degrees of freedom) was used and accounted for the full complexity of the Earth (wave-speed structure, density, topography, and bathymetry). This experiment required 2.5 TB of memory, relied on an MPI implementation that obtained a 99.3% vectorization ratio, and ran using 5 teraflops (30%

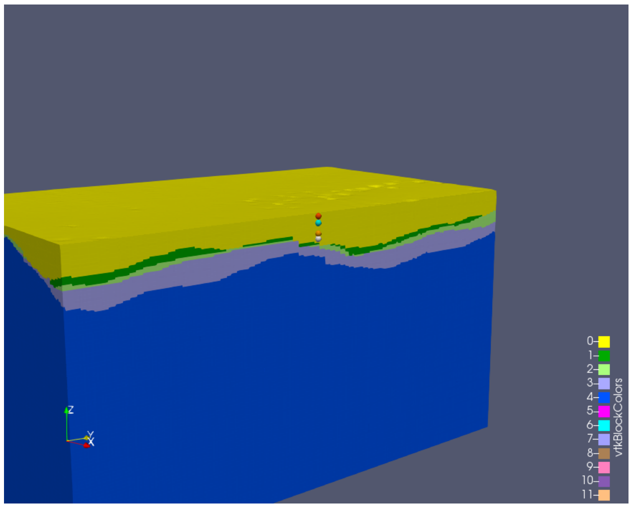


Figure 1.1: A cross-section of the Dry Alluvium Geology (DAG) Site. Different colors represent different geologic units. In the middle, 4 dots of different colors represent the 4 explosion experiments. Credit to Carène Larmat.

of max computing speed). High resolution computations ran in as low as 5 seconds.

The rest of this paper is organized as follows. In Section 2, we describe the DAG site of the Source Physics Experiment, describing the source, distributions of stations, and the various experiments. Section 3 is an overview of our process and methods. Section 4 describes results with a short a discussion for each experiment. We summarize our general conclusions and give an answer to the question of the paper in Section 5. Finally, we end with further work in Section 6.

1.2 The Source Physics Experiment

1.2.1 The DAG Site

The Dry Alluvium Geology, hereafter referred to as DAG, site is within the Nevada National Security Site. It consists of a borehole in which multiple explosions have been conducted at different depths and intensities. The geology surrounding the borehole consists of five distinct geological units. For the sake of our numerical experiments, we group these 5 units into 3 layers: Alluvium (unit 1), Tuff (unit 2,3,4), and Paleozoic (unit 5) layers. A geological survey along with numerous borehole core analysis has determined the average and variability of the elastic properties of each of the geologic units. This has allowed for the entire environment to be modeled as seen in figure 1.1. Due to uncertainty about elastic properties throughout the entire geologic medium, we create experiments varying the elastic properties of the medium. These are known as *velocity models*. The following table shows the measured elastic properties of the DAG site, specifically the velocity properties of each unit. This velocity model will be the reference, or control, for our experiments. All other variations of velocity models are relative to this case. These elastic properties are varied and used as separate experiments. The V_p, or velocity at which the P-wave travels through the rock, was varied and the V_s, or S-wave velocity, changed proportionally to preserve the V_p/V_s ratio for each unit.

unit	P-wave velocity (m/s)	S-wave velocity (m/s)
1	1600	950
2	1982	1040
3	1937	1015
4	2430	1270
5	4967	2365

Table 1.1: Measured Elastic Properties of the DAG Site. The control experiment known as case 0.

The variations provided in table 1.2 are applied to the elastic properties in table 1.1 and are recorded as separate cases.

	Velocity Change of Medium (%)		
case	Alluvium	Tuff	Paleozoic
0	0	0	0
1	-30	-30	+30
2	-30	+30	+30
3	-30	+30	-30
4	+30	+30	-30
5	+30	0	-30
6	+30	0	+30
7	homogeneous medium		

Table 1.2: Change in percent of the V_p elastic property for each mesh case from the base case, mesh case 0. The layers in table 1.1 were changed accordingly.

1.2.2 The Centroid Moment Tensor

The Centroid Moment Tensor (CMT) is the central object describing the source of a seismic event. It is an example of a Cauchy stress tensor that describes the direction of shearing and stretching forces in 3 dimensions. This information is captured in a 3×3 matrix, $\mathbf{M}$, which is symmetric, i.e. $\mathbf{M} = \mathbf{M}^T$.

$$\mathbf{M} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{12} & M_{22} & M_{23} \\ M_{13} & M_{23} & M_{33} \end{bmatrix}$$

We describe explosions using non-zero diagonal entries, representing expansion forces with the same amplitude. Earthquakes occur when friction between tectonic plates is overcome, resulting in a shearing effect along the *fault plane*. Earthquake sources are represented as non-zero off-diagonal entries of the CMT.

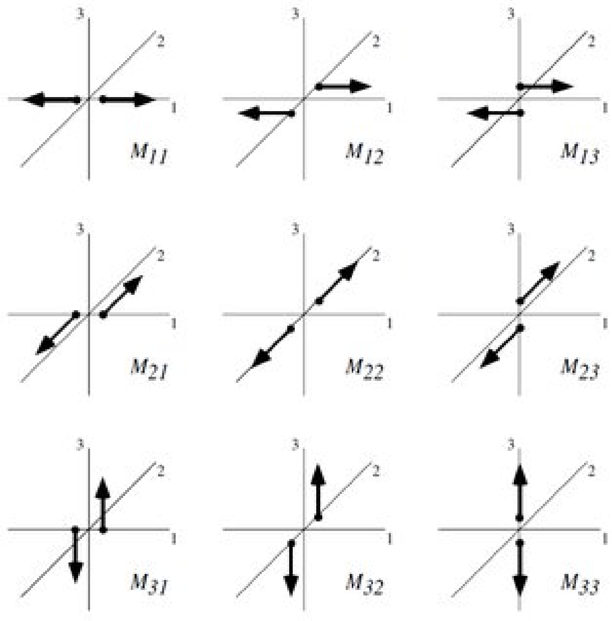


Figure 1.2: Description of expansion and shearing forces in different directions. Linear combinations of these describe arbitrary stress/strain forces. Image source comes from [Caputa et al. (2015)].

The intensity of an earthquake or explosion, known as the *seismic moment* and denoted by M_0, can be computed using the formula

$$\mathrm{M}_0 = \frac{1}{\sqrt{2}}(\mathrm{M} : \mathrm{M})^{1/2}$$

where the : is the double dot product from dyadic tensor algebra (also known as the Frobenius 2-norm).
The four explosions conducted at the DAG site were modeled in the mesh case 0.

DAG Explosion	Depth (m)
`DAG-1`	385
`DAG-2`	299.8
`DAG-3`	149.9
`DAG-4`	51.6

Table 1.3: Depths of each DAG site explosion.

1.2.3 Distribution of Stations

The DAG explosions were recorded on a network of seismic stations situated around the explosion borehole up to 4 km away. As seen in figure 1.3, the density of stations is significantly higher to the south of the borehole head than to the north. These stations record the seismic activity of the ground in three coordinates: the North, the East, and depth-wise.

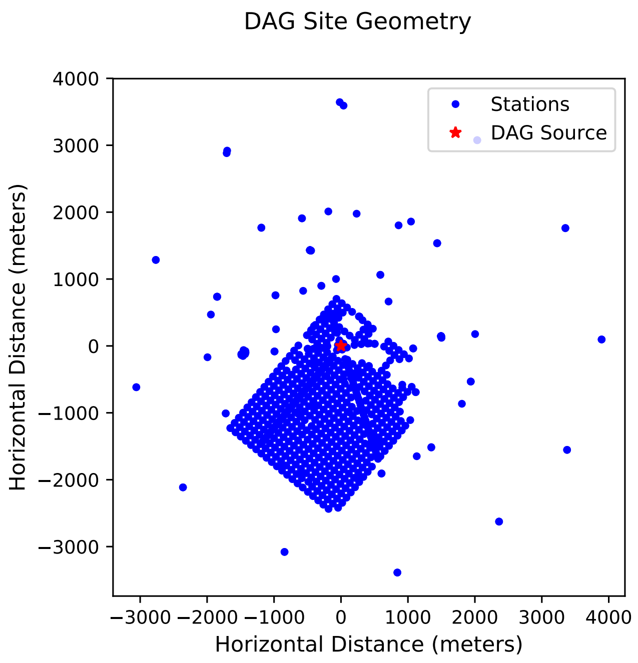


Figure 1.3: The DAG site station geometry, stations are mapped with blue dots and the DAG borehole head is denoted with a red star. Note the heterogeneous station distribution, with a higher density south of the DAG site than north.

1.2.4 Experiments

Mesh Experiments

The first set of experiments consist of varied the elastic properties of the surrounding layers of Earth. Meshes corresponding to the 8 different cases presented in 1.2 were generated, using the DAG-2 explosion as the source. We vary the V_p and V_s in the mesh cases as to preserve the ratio in each unit. The experiment names are `DAG2_case0` all the way to `DAG2_case7`. See 1.2.

Depth Experiments

The next set of experiments comprised of changing the source depth of the explosion and keeping the control mesh case 0 the same. The experiment names are `DAG1_case0`, `DAG2_case0`, `DAG3_case0`, and `DAG4_case0`.

Random perturbation Experiments

Our next experiment aims to a better description of the Earth model as elastic properties within each geologic unit varies randomly due to complexity of Earth processes. We choose to vary randomly the elastic properties of the Alluvium layer. We design tomographic files allowing for non-constant elastic properties. This was done by associating density, V_p, and V_s wave velocities to each point of the model thanks to a “tomography” file, according to the format given in the SPECFEM3D manual [Komatitsch et al. (2018)]. A "tomography" model is a 3D grid of points for which each of the properties are given and from which elastic properties for any point is derived by linear interpolation.

We vary the amplitude of the reference velocity ($V_p = 1600$ m/s) by either 10% or 30% and uniformly assign velocities within these amplitudes to each point in the mesh. The solver linearly interpolates which points belong to each layer. Then a Gaussian filter was applied with three predetermined correlation lengths in the three Cartesian coordinates. This allows for control on the smoothness of variation between velocity changes. Table 1.4 describes the random tomography experiments. Each case name is named using the format `rt_Ac`, where `rt` stands for "random tomography", $A \in \{1, 2, 3\}$, and $c \in \{a, b\}$.

Case name	Correlation Length	Velocity perturbation (%)
`rt_1a`	500/500/500	10
`rt_1b`	500/500/500	30
`rt_2a`	500/500/60	10
`rt_2b`	500/500/60	30
`rt_3a`	60/60/60	10
`rt_3b`	60/60/60	30

Table 1.4: Random tomography experiments

Earthquake Experiments

Lastly, CMTs of equal seismic moment to that of the DAG-2 explosion that had shearing effects instead of expansion effects were designed. This was done by placing a factor of the seismic moment, using the seismic intensity formula, in each of the off-diagonal entries, resulting in shearing effects in any of three directions. These experiments were named the `DAG2_case0_EQ_tp`, `DAG2_case0_EQ_rp`, and `DAG2_case0_EQ_rt`, where r, t, and p are the radial, tangential, and depth coordinates, respectively.

Table 1.5 contains all the experiments conducted in this work.

List of Experiments		
`DAG-2_case0`	`DAG-2_rt_1a`	`DAG-1_case0`
`DAG-2_case1`	`DAG-2_rt_1b`	`DAG-3_case0`
`DAG-2_case2`	`DAG-2_rt_2a`	`DAG-4_case0`
`DAG-2_case3`	`DAG-2_rt_2b`	`DAG-2_case0_EQ_tp`
`DAG-2_case4`	`DAG-2_rt_3a`	`DAG-2_case0_EQ_rp`
`DAG-2_case5`	`DAG-2_rt_3b`	`DAG-2_case0_EQ_rt`
`DAG-2_case6`		
`DAG-2_case7`		

Table 1.5: All experiments conducted

1.3 Methods

1.3.1 SPECFEM3D

The open-source software, SPECFEM3D, is used to simulate seismic wave propagation using full waveform imaging based on finite and spectral element methods using a continuous Galerkin technique [Komatitsch et al. (2018)]. Spectral element method is a special case of the finite element method based on higher-order elements.

Each experiment was given its own folder labeled with one of the names in table 1.5. The process consisted of placing the appropriate configuration files that determine the scenario of the experiment. Each folder consisted of a `DATA` and `MESH` folder. The `DATA` folder contained a file of simulation parameters, a list of stations (and their locations), a 3D velocity model file, and the CMT source file. The `MESH` folder contained files describing the material and elastic properties of the mesh.

SPECFEM3D requires the 3D velocity format in a specific format. This information was stored in databases that needed to be generated. Databases were generated for each of the mesh cases in table 1.2 and for each of the random tomography experiments (as they relied on a random tomography file) in table 1.4. Generating databases created an `OUTPUT_FILES` folder that would eventually contain the unprocessed seismic data.

To generate data, the parameter file had to be configured to refer to the correct database. Data that was generated on Grizzly CPU cluster and took on average 6 to 8 hours to generate. Jobs on Grizzly required 35 nodes. Data that was generated using the Kodiak hybrid-architecture CPU/GPU cluster would take less than an hour to generate. Jobs on Kodiak required 18 nodes. We found that the 12-20 times speed up in data generation between CPU and GPU clusters was consistent with the results of [Komatitsch et al. (2010)].

Noise

Throughout the study, we were comparing the results of our reference case `DAG2_case0` to results generated previously by Carène Larmat. We noted a significant increase in numerical noise in our waveforms even after filtering. We suspect this noise is due to either the use of different compilers or the varying architectures used to run the simulations. This is, as of now, a subject of further investigation.

1.3.2 Post-Processing

SPECFEM3D produces waveform data for each station in three components: eastern component (E), northern component (N), and vertical component (Z). This raw data was converted to the common format used in seismology, the `SAC` format. Then the `.sac` files were passed through a low pass filter only allowing frequencies up to the resolution frequency of the solver. Lastly, the low-pass filtered data was rotated to produce radial (R) and transverse (T) components. Subsequent low-pass filtering was done as necessary through python.

1.3.3 Analysis

The analysis for this project was conducted using Python. The Python module Obspy, a seismology framework for python, was the primary tool for data input/output. Obspy stores seismic data in a ”stream”. Individual waveform data is stored in a ”trace” and carries attributes such as the station name, distance from the seismic event focus, latitude and longitude... etc. The modules Numpy and Matplotlib were used for calculation and plot generation, respectively.

Waveform filtering

Seismic data was low-pass filtered directly after being read into the stream. This low-pass filter is built into Obspy, and removes all the frequencies higher than a cut off frequency, which is specified by the user. The cut off frequencies used for the cases considered are presented in table 1.6, and were an integer value below maximum frequencies allowed in each mesh case.

Case	Max Frequency (Hz)
DAG-2 case 0	24
DAG-2 case 1	17
DAG-2 case 2	17
DAG-2 case 3	17
DAG-2 case 4	32
DAG-2 case 5	27
DAG-2 case 6	27
DAG-2 case 7	26
DAG-1 case 0	24
DAG-3 case 0	24
DAG-4 case 0	24
All DAG-2 rt cases	24

Table 1.6: Low pass filtering cut off frequency for the DAG cases considered.

Visualization of Geometry

We visualize the geometry of the DAG site by plotting the stations and DAG source based of the attributes of horizontal distance from the borehole head. The geometry is plotted in figure 1.3.

Waveform plotting

For this visualization, we sort the stations based on angular direction from the DAG source. For each group of stations, the corresponding waveforms are plotted as a function of distance alongside of constant move out lines of 1, 1.5, and 2 km/s. Angular increments of 45 degrees are used to provide sufficient station density. The geometry of the stations whose waveforms have been selected are also plotted for reference.

Arrival times

Arrival times are picked out using classic STA/LTA, a waveform processing algorithm built into Obspy. Classic STA/LTA is a trigger picking algorithm that considers a short-time window (STA) and a long-time window (LTA) of a waveform, e.g. $STA = 6$ frames, $LTA = 140$ frames. Then, a characteristic function of the waveform is calculated by considering the signal average of both windows and taking the ratio, hence STA/LTA. An event is detected when the STA/LTA ratio is above a certain value, the on and off thresholds are set manually. An example trigger and characteristic function may be seen in figure 1.4. Classic STA/LTA works very well on idealized seismic waveforms, where P and S arrivals are clearly separated and there is no noise between the respective waves. However, lots of noise was present in our waveforms resulting in noisy

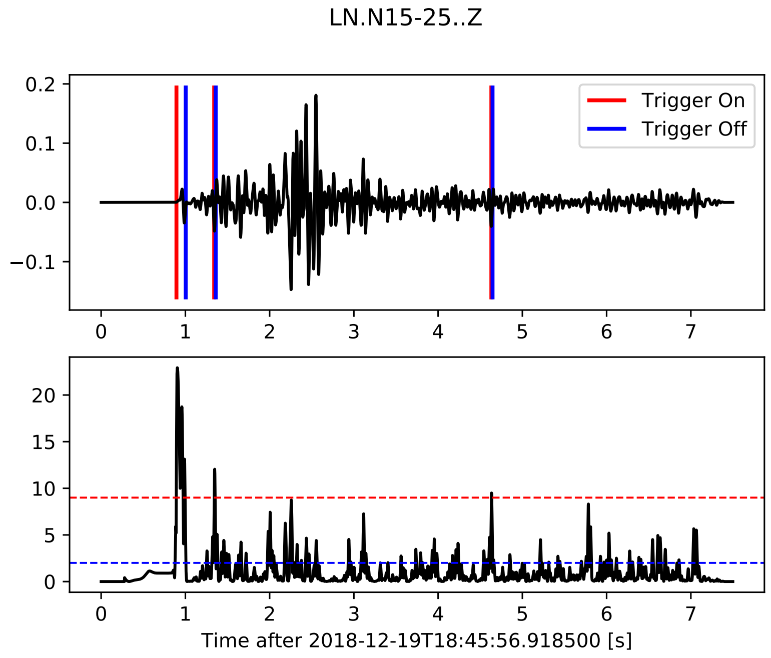


Figure 1.4: An example waveform (top) and characteristic function (bottom) calculated using the classic STA/LTA algorithm. The red and blue dotted lines indicate trigger on and off, respectively. Note the false triggers occurring approximately at $t = 4.75$ seconds and $t = 1.3$ seconds due to the numerical noise present in the waveform. Here, the on and off trigger values have been set to 9 and 2, respectively.

characteristic functions. This resulted in false triggering, an example of which may be seen in figure 1.4. Waveforms were plotted as a function of distance alongside of the their arrival times to manually check for false triggers, as in figure 1.5. With approximately 720 waveforms to check per experiment, this proved to be a time consuming task, as the classic STA/LTA parameters needed to be varied for each of the 20 experiments. Additionally, waveform noisiness varied considerably with distance, resulting in false triggers when solely one set of classic STA/LTA parameters was used for the entire $0 - 4$ km range. In the end, classic STA/LTA parameters were tweaked based on 7-9 different distance increments outlined in table 1.7 for DAG-2_case0.

DAG-2_case0 classic STA/LTA parameters				
Distance interval (meters)	STA (frames)	LTA (frames)	Trigger on	Trigger off
0-200	2	40	9	2
200-400	3	85	6.5	2
400-800	6	108	9.5	2
800-1800	6	130	8	2
1800-2300	6	140	8.5	2
2300-3000	6	130	8	2
3000-4000	6	140	8.6	2

Table 1.7: Classic STA/LTA parameters used for DAG-2_case0.

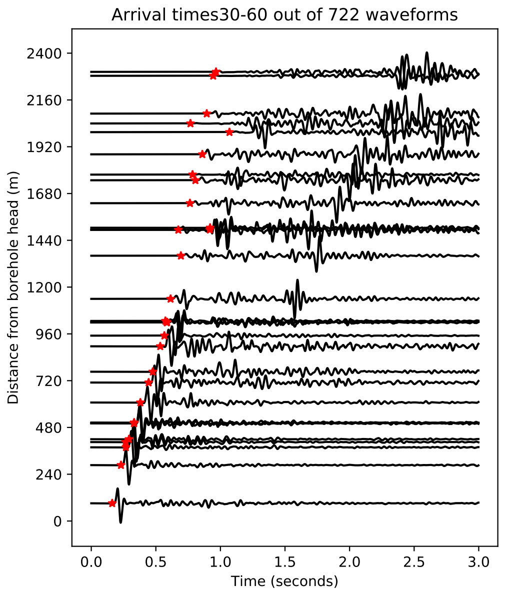


Figure 1.5: The plots generated for manual verification of arrival times. Waveforms are plotted in black as a function of distance, while arrival times are the red stars.

Amplitude ratios

First, we analyze the amplitudes of waveforms over the whole DAG site. We plot this amplitude as a "heat map" over the station geometry to investigate where the maximum amplitudes are achieved. Second, in order to process the amplitude ratios of our S and P waves, we build off of our arrival times. We considered three different P wave windows, $[t_{pick}, t_{pick} + 0.15]$, $[t_{pick}, t_{pick} + 0.1]$, and $[t_{pick}, t_{pick} + t_{pick} * v_p/v_s]$. Ultimately, data using the third window was selected after manually inspecting the P wave window for quality. The S-wave window was picked consistently as $[t_{pick} * v_p/v_s, 2 * t_{pick} * v_p/v_s]$. An example of the P and S wave windows is provided in figure 1.6.

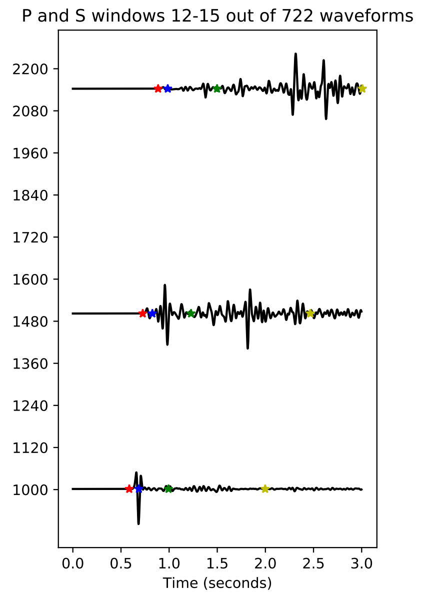


Figure 1.6: Waveforms plotted as a function of distance with their corresponding P and S wave windows highlighted. For this particular example, the $[t_{pick}, t_{pick} + 0.15]$ P wave window is used and demarcated with red and blue stars, and the $[t_{pick} * v_p/v_s, 2 * t_{pick} * v_p/v_s]$ S window is used and demarcated with green and yellow stars.

Using these windows we analyze the P wave window using four metrics that characterize energy distribution in waveforms. The first is the $\max(T)/\max(Z)$ metric, in which we consider the maximum amplitude achieved in the transverse component (T) of the P wave window, and divide it by the maximum amplitude achieved by the vertical component (Z). The second is the $\max(T)/\max(\sqrt{R^2 + Z^2})$ metric, in which a similar procedure is applied to the ratio of the maximum of the T component, and the maximum of the square root of the vertical (Z) and radial (R) components. The third metric is $\mathrm{rms}(T)/\mathrm{rms}(Z)$, in which the ratio of the root mean squared of the T and Z components are considered. The final metric is $\mathrm{rms}(T)/\mathrm{rms}(\sqrt{R^2 + Z^2})$, where we consider the root mean square of the T component in proportion to the root mean square of the L2 norm of the R and Z components. The goal of all these metrics is to analyze which components, Z, T, or R, dominate the P wave window. As the energy of a seismic wave may be loosely attributed to its amplitude, these metrics effectively measure the ratio of the energies present in the P wave components. Note that the last metric is a slight generalization of the third metric when the R component is negligible. Following some investigation, it was discovered that the metrics utilizing the root mean square of the windows provided better results than their max counterparts, and so this metric was preferred. P/S ratio analysis was done by considering $\mathrm{rms}(\sqrt{Z^2 + T^2 + R^2})$ for both the P and S wave windows, and taking the ratio.

1.4 Results and Discussion

1.4.1 Explosions vs Earthquakes

In this section, we consider two different metrics to separate earthquakes from explosions. We consider the effect of mesh properties, specifically the variation of P and S wave velocities, on our ability to discern earthquakes from explosions. Experiments considered in this section can be found in table 1.8.

List of Experiments	
Explosions	Earthquakes
DAG-2 case 0	DAG-2 case 0 EQ tp
DAG-2 case 1	DAG-2 case 0 EQ rp
DAG-2 case 2	DAG-2 case 0 EQ rt
DAG-2 case 3	
DAG-2 case 4	
DAG-2 case 5	
DAG-2 case 6	
DAG-2 case 7	

Table 1.8: Explosion and earthquake cases considered for comparison.

As a preliminary result, we consider the earthquake, `DAG2_case0_EQ_tp`, and the explosion, `DAG2_case0`, with the same seismic moment and depth. The surrounding mesh properties in the two experiments did not change. We use the $\log(P/S)$ metric to visualize the overall trend in the data. As seen in figure 1.7, $\log(P/S)$ discerns earthquake from explosion well for distances smaller than 2 km, with the explosion achieving consistently higher values of $log(P/S)$ than the earthquake. The explosion's high $log(P/S)$ values are indicative of the P wave amplitude dominance that we expect from explosions, as it is a compression event. In comparison, the values of $\log(P/S)$ achieved by the earthquake demonstrate a strong S wave amplitude dominance originating from the shearing nature of the earthquake. The downtrend of the explosion data beginning at around 1 km is attributed to S wave generation. This phenomenon is the result of P waves interfering with other seismic waves and geologic interfaces. Stations further from the source see more S wave generation, as this effect compounds over time. The lack of information in the $2 - 4$ km range is due to the lack of stations present at the DAG site, and will be the subject of further investigation.

Our next result is a comparison of all explosion mesh cases from table 1.2 to all earthquake cases as considered in figure 1.8. Individual cases are colored for understanding in figure 1.9. Until around 1 km, we still have good separation between all of the earthquake and explosion data. However, for distances further than 1 km, certain explosion $\log(P/S)$ ratios begin to blend with the earthquake ratios. As demonstrated in figure 1.10, the blending primarily occurs from DAG-2 explosion data using mesh cases 1, 2, and 3. As presented in table 1.2 cases 1, 2, and 3 are cases in which the P wave velocity was designed to travel 30% slower than the control case in the Alluvium layer. This reduction in velocity provides more time for seismic wave interference from propagating in the complex medium, as waves interact with the interfaces between geologic layers through reflection

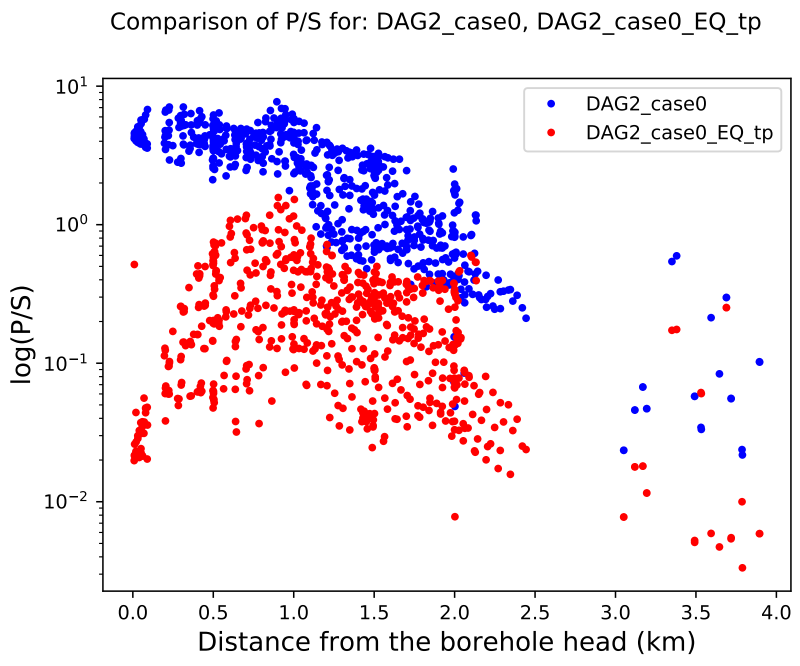


Figure 1.7: Comparison of earthquake, labeled in red as DAG-2_case0_EQ_tp, and explosion, labeled in blue as DAG-2_case0. Earthquake and explosion are distinctly separated with a little bit of mixing at and beyond 2 km.

and transmission. These interactions generate S waves, resulting in the decreasing $\log(P/S)$ ratios at stations further away from the borehole.

In an attempt to find other earthquake and explosion distinguishing metrics, we also examined the P wave for dominating components. The chosen metric is $\text{rms}(T)/\,\text{rms}(Z)$, where we consider the root mean square of the transverse (T), and vertical (Z) components. The root mean square corresponds loosely to computing the "energy" of the components. Effectively, we are measuring the ratio of energies between components of the P wave. Results are presented in figure 1.11. For distances less than 500 meters, the earthquake P wave window is dominated by Z component, while the explosion's P wave window is dominated by the T component. However, both seismic events become indistinguishable in this metric for distances greater than 500 meters the mixing of the earthquake and explosion data.

1.4.2 Variations of Explosions

Mesh Velocity Variation

As discussed in the previous section, varying the mesh P wave velocities for DAG-2 cases 0-7 achieved a wide range of $\log(P/S)$ behavior. We further investigate this in figure 1.12. The homogeneous mesh, case 7, demonstrates high P wave amplitude preservation with the largest values of $\log(P/S)$ for any case. The reference case, case 0, sits between the faster and slower Alluvium P wave velocity groups. Cases 1, 2, and 3, for which the velocity of the P wave has been

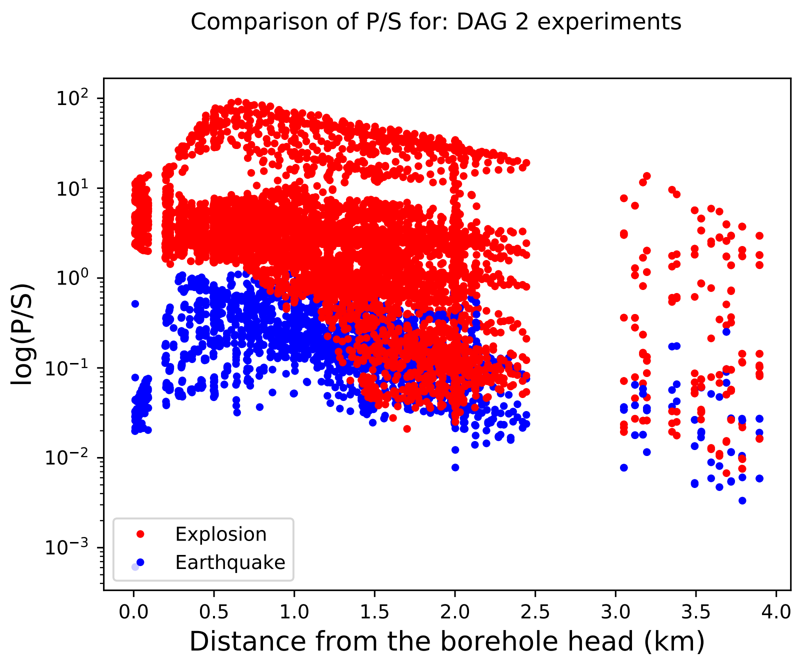


Figure 1.8: Comparison of all the DAG-2 explosion mesh cases and all the earthquake cases. Earthquakes are demarcated in blue, and explosions in red. Note the blending of earthquake data with some of the explosion data for distances upwards of one kilometer.

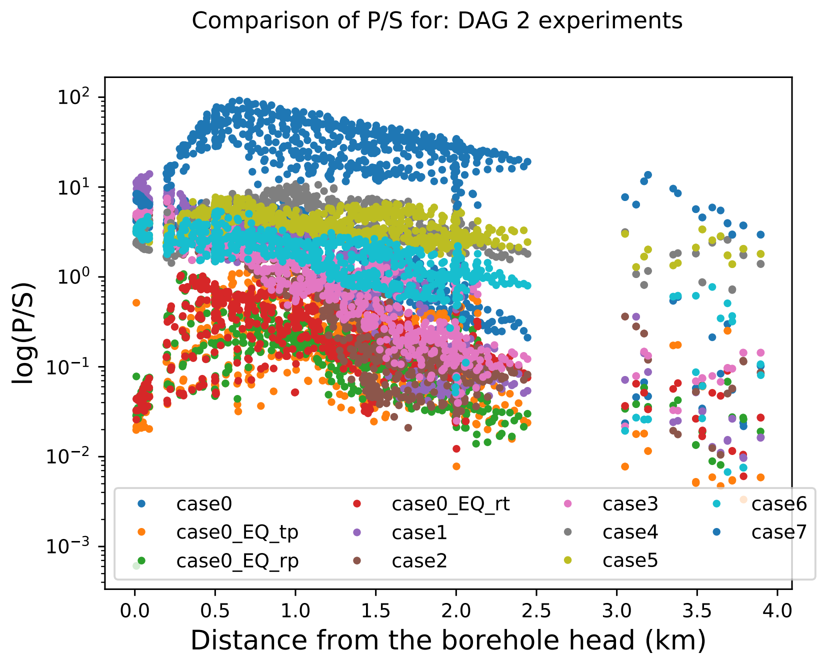


Figure 1.9: Comparison of all the DAG-2 explosion mesh cases and all the earthquake cases. Note the blending of earthquake data with some of the explosion data for distances greater than 1 km.

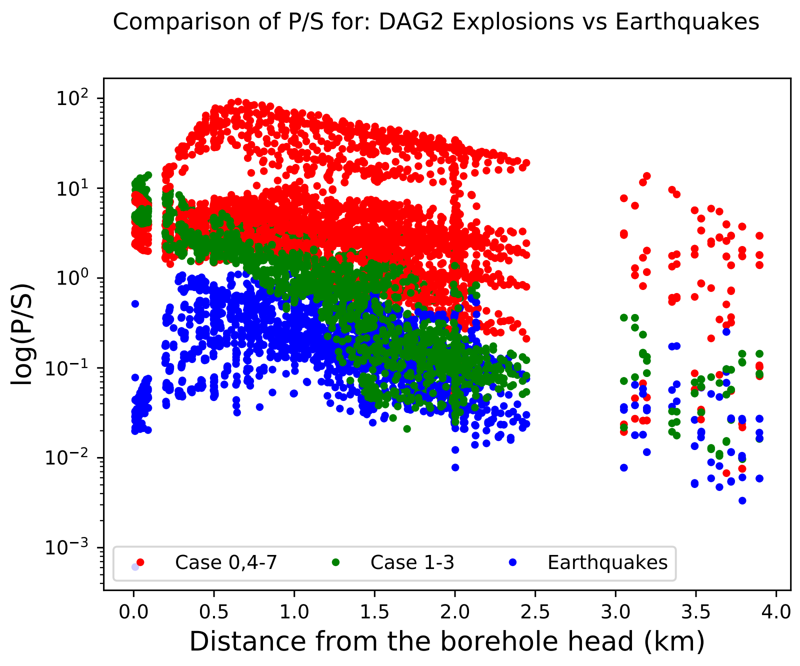


Figure 1.10: Comparison of all the DAG-2 explosion mesh cases and all the earthquake cases. Explosion mesh cases 1-3 have been colored differently for clarity. For cases 0, 4-7, the explosions are well separated from the earthquakes.

manually slowed by 30% in the Alluvium layer, has the highest amount of S wave generation, seen in the sharp drop in values of $\log(P/S)$ as distance from the borehole increases. For cases 4, 5, and 6, with Alluvium P wave velocities 30% faster than the control case, S wave generation is not as prevalent. The varying velocities may be seen in figure 1.13, in which, for short distances, the explosions with slowed Alluvium P wave velocities arrive later than those than their faster traveling counterparts.

Explosion Depth Variation

We have considered explosions at a variety of depths, as seen in table 1.3. Arrival time results are shown in figure 1.14. For distances less than 1 km, the shallower explosions, DAG-4 and DAG-3, arrive first. This corresponds to the proximity of these explosions to the stations on the surface of the earth. However, for distances greater than 1 km, the deepest explosion arrives earliest. Explosions at increasing depths send seismic waves into the deeper geological layers of the DAG site, which propagate the P and S waves faster, allowing for these earlier arrival times. Figure 1.15 provides the effect of depth on $\log(P/S)$. For distances greater than 1 km, the deeper explosions, DAG-1 and DAG-2, both have sharp downward trends of $\log(P/S)$ values as a function of distance, indicative of S wave generation. This behavior is in contrast to the shallower explosions, DAG-3 and DAG-4, in which P wave amplitudes are preserved, as the $\log(P/S)$ downtrend is not nearly as steep.

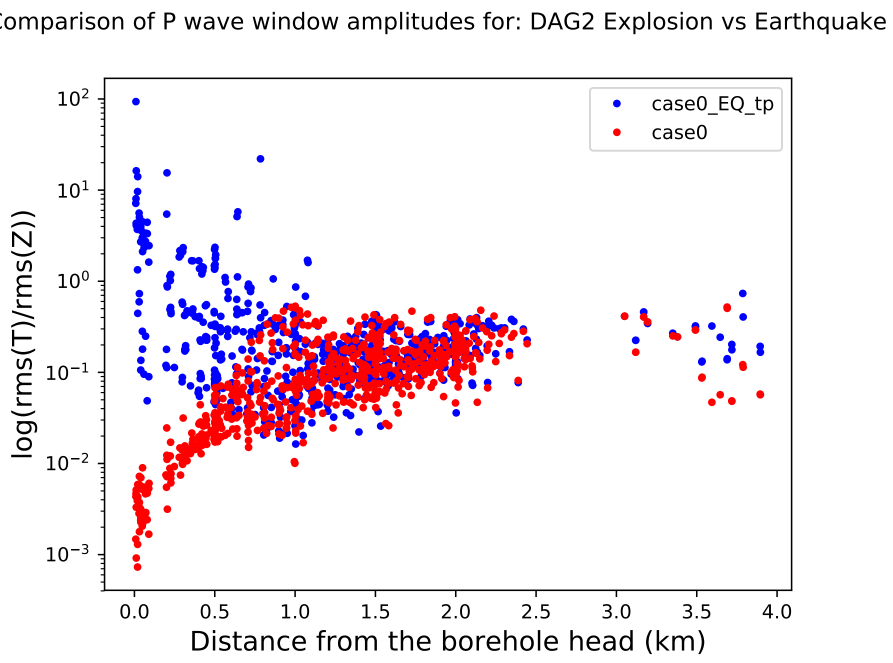


Figure 1.11: Comparison of the wave components present in the earthquake, DAG-2_case0_EQ_tp, and explosion, DAG-2_case0. For distances less than 500 meters, the earthquake and explosion are distinguished by dominating vertical (Z) and transverse (T) components, respectively. For distances greater than 500 meters the events indistinguishable.

1.4.3 Random Perturbation Results

Here, we describe the results of the random tomography experiments. This is done in accordance with table 1.4. Results are seen in figures 1.16 and 1.17. Neither the amplitude nor the correlation length seems to have an effect on the $\log(P/S)$ metric. This is, as of the writing of this report, a subject of further investigation.

1.4.4 Wave Propagation in a 3D Medium

Due to topographical variation of the geologic layers that comprise the DAG site, waveform structure varies considerably from location to location. This variation is demonstrated in figure 1.18 and 1.19. For stations west southwest of the DAG source, at distances greater than 1500 meters, P waves arrive consistently to the left of the 1 km/s move-out line. This is in stark contrast to the east southeastern stations, for which all P wave arrivals stay consistently on the right side of the 1 km/s move-out line. This variation of arrival times as a function of azimuth is responsible for the splitting in arrival times after 1 km, seen in figure 1.20.

Topological variation of the DAG site geology further manifests itself in the component characteristics of the waveforms. As seen in figure 1.21, the south southwestern and east southeastern corners of the DAG site exhibit transverse (T) component dominance in the P wave window, while vertical (Z) and radial (R) components are dominant in other areas. This broad component vari-

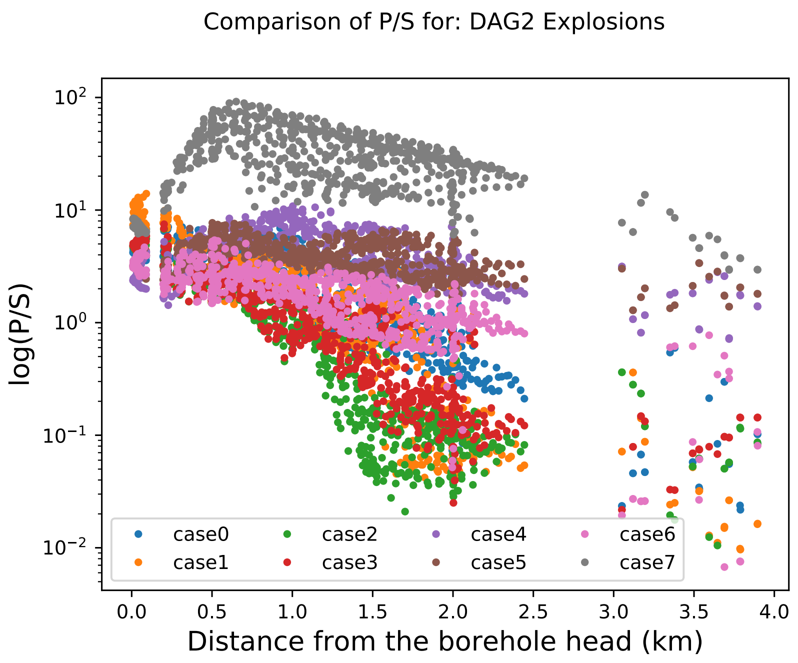


Figure 1.12: Comparison of the different DAG-2 explosion cases using the $\log(P/S)$ metric. The high values achieved by the homogeneous mesh, case 7, is indicative of P wave preservation. A large amount of S wave generation is seen in the decreasing values of $\log(P/S)$ for cases 1, 2, and 3, whereas P wave preservation is dominant in the cases 4, 5, and 6. The reference case, case0, sits between both the aforementioned groups.

ation is attributed to the geological layers interacting with the components of the seismic waves, propagating some and transforming others depending on the surrounding topography and geology.

To conclude our results, we consider 3 earthquakes at the same depth and magnitude and vary which off diagonal component of the centroid moment tensor is non-zero. As observed in figure 1.22, even with this minute variation in source shearing, the three seismic events have very different $\log(P/S)$ signatures. This is testament to the inherent complexity of wave propagation in a non-homogeneous 3-D medium, which amplifies even slight variations in initial conditions.

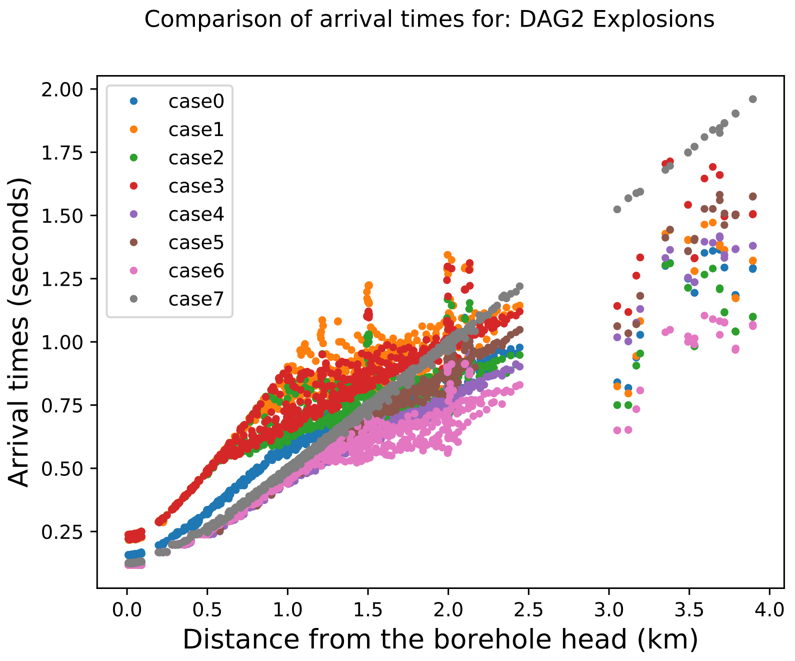


Figure 1.13: Comparison of the arrival times of the DAG-2 mesh cases. Note the mesh variations with slower P wave velocity in the alluvium layer, cases 1, 2, and 3, arrive later than the other mesh cases for short distances.

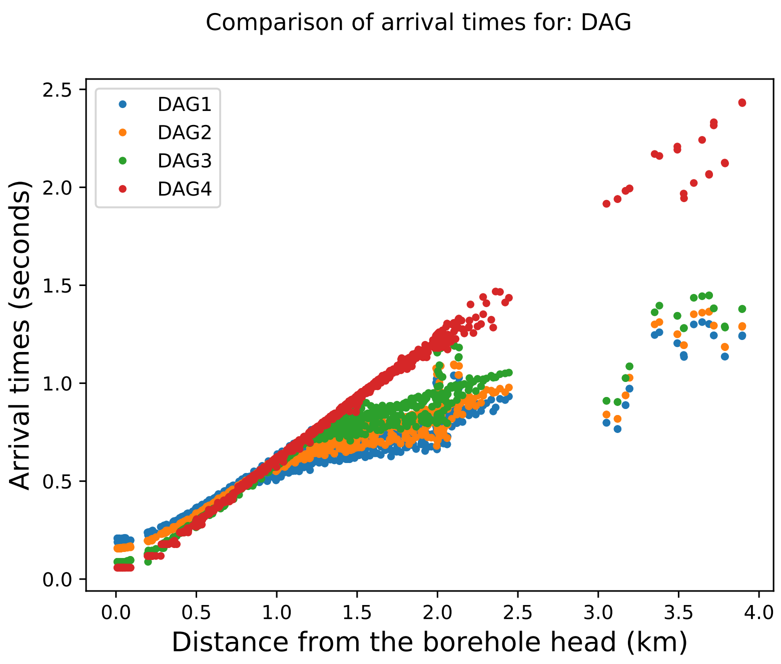


Figure 1.14: Comparison of the arrival times of DAG explosions in ascending depth from DAG-1 to DAG-4. The shallowest explosion, DAG-4, arrives first for distances less than 1 km. For distances further than 1 km, the deeper explosions propagate waves into the lower geological layers in which both P and S waves travel faster, resulting in faster arrival times.

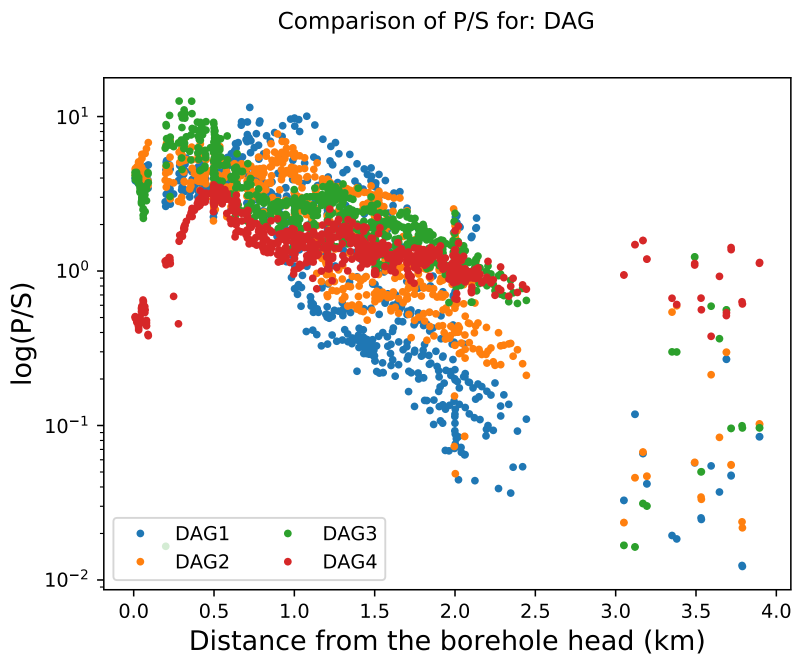


Figure 1.15: Comparison of DAG explosions in decreasing depths from DAG-1 to DAG-4. The strong downtrend in $\log(P/S)$ values for deeper explosions is indicative of high S wave generation. This behavior is not as drastic in shallower explosions.

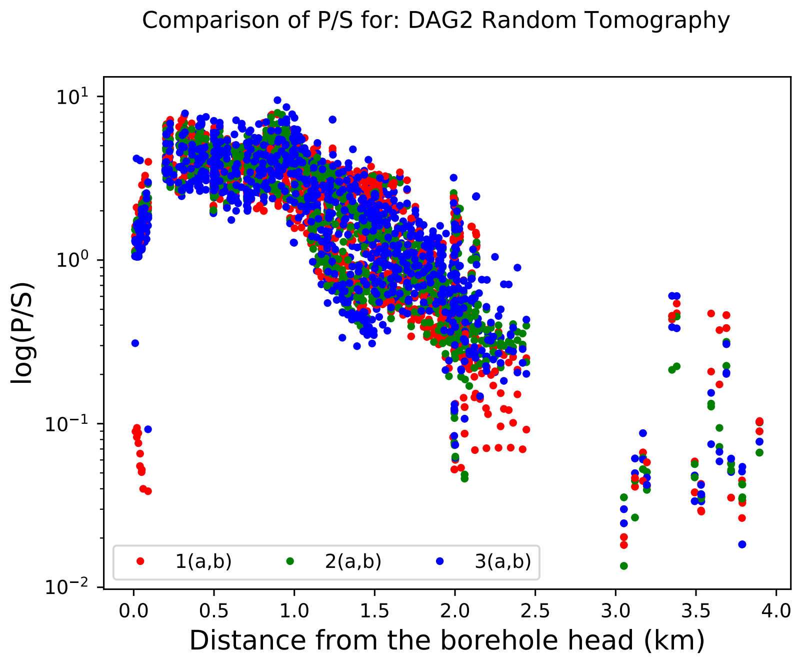


Figure 1.16: Comparison of random tomography mesh cases, colored based on correlation length. Red corresponds to lengths of $500/500/500$, green to $500/500/60$, and blue to $60, 60, 60$.

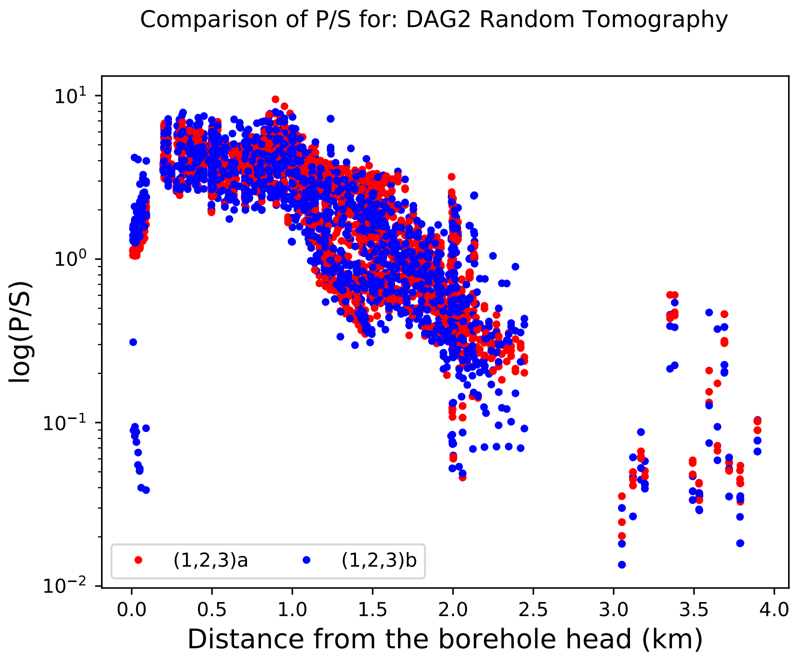


Figure 1.17: Comparison of random tomography mesh cases, colored based on velocity amplitude fluctuations. Red corresponds to a variation of 10%, and blue to a variation of 30%.

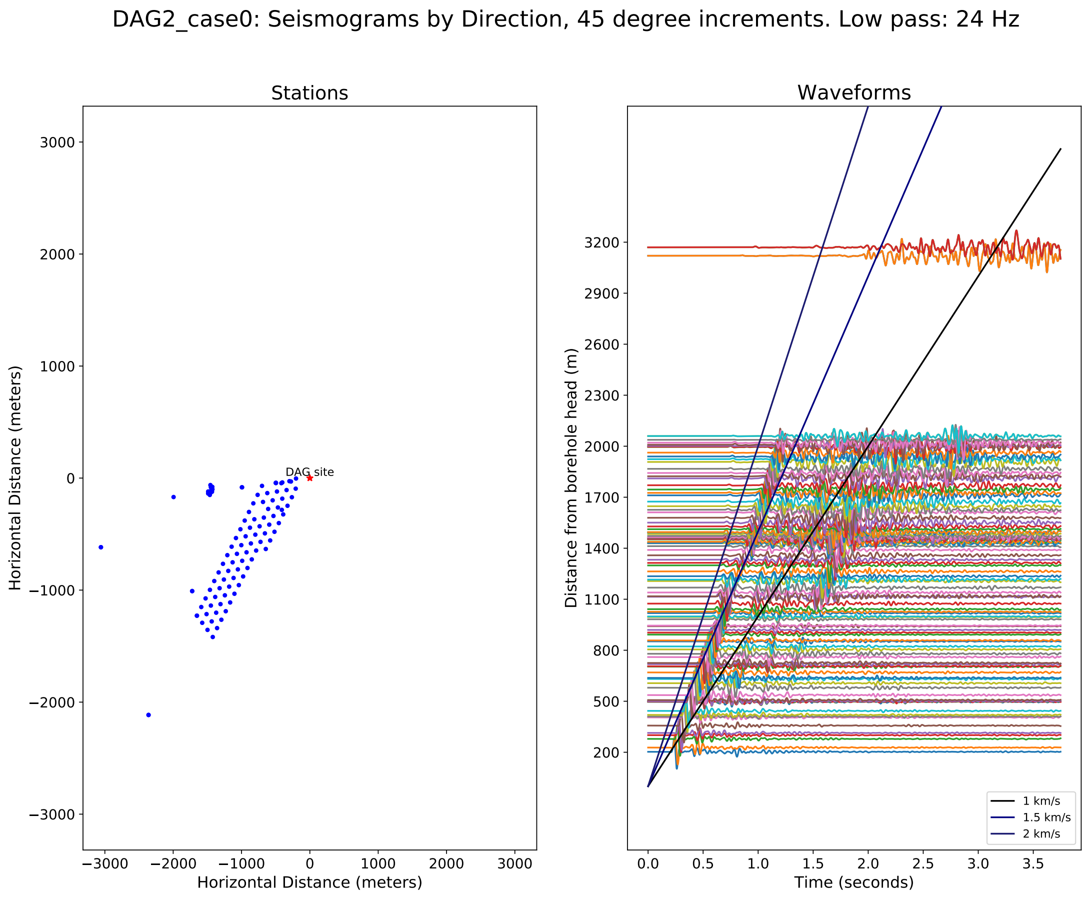


Figure 1.18: Waveforms of DAG-2 case 0 stations west southwest of the DAG source as a function of distance from the borehole head. For distances greater than 1500 meters, the P waves stay to the left of the 1 km/s move out line.

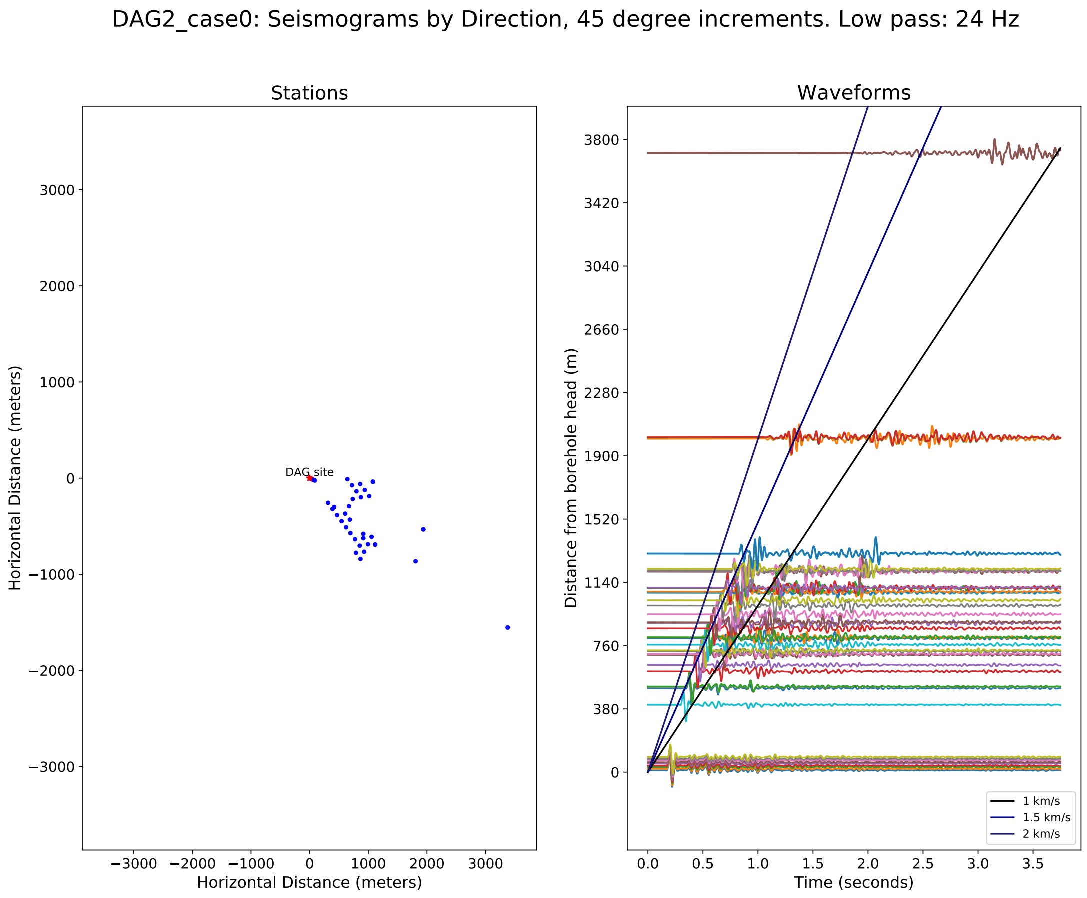


Figure 1.19: Waveforms of DAG-2 case 0 stations east southeast of the DAG source as a function of distance from the borehole head. The P waves stay to the right of the 1 km/s move out line.

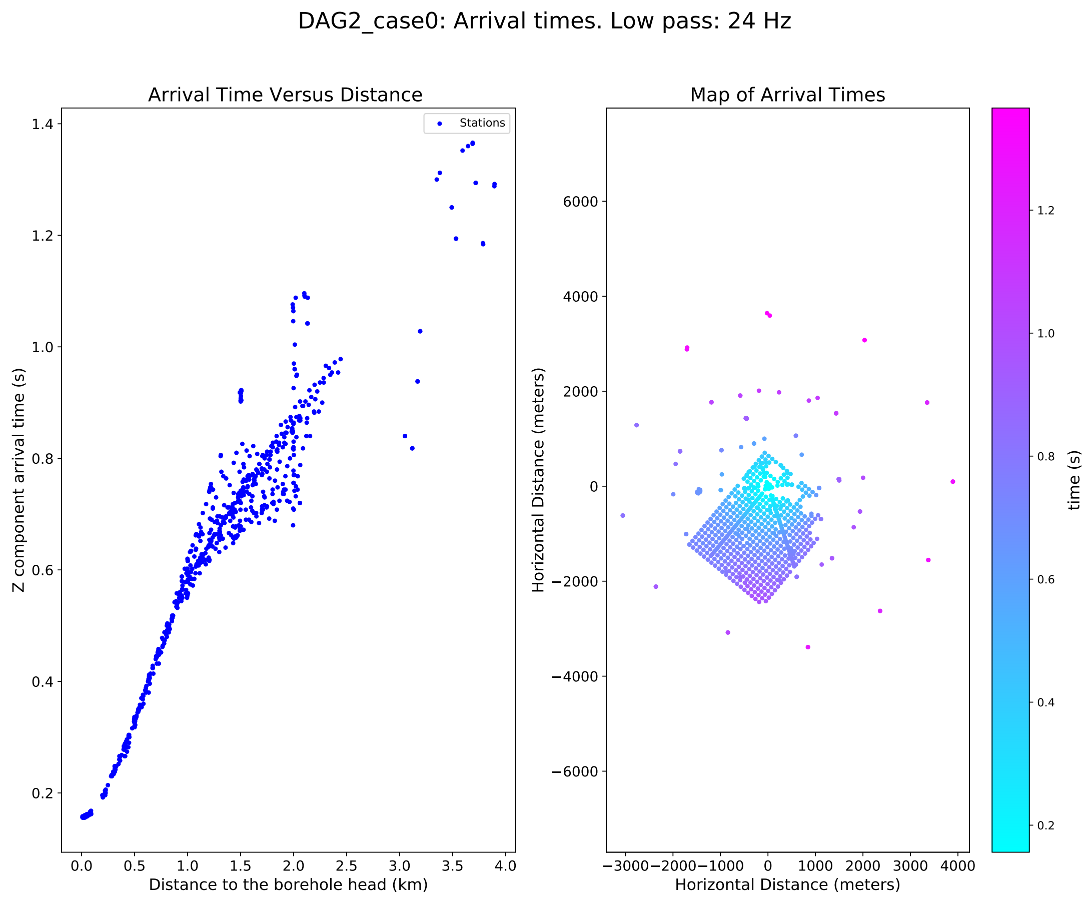


Figure 1.20: Arrival times for DAG-2 case 0. The splitting beginning at around 1 km is attributed to the topological variation of DAG geology in different directions from the source.

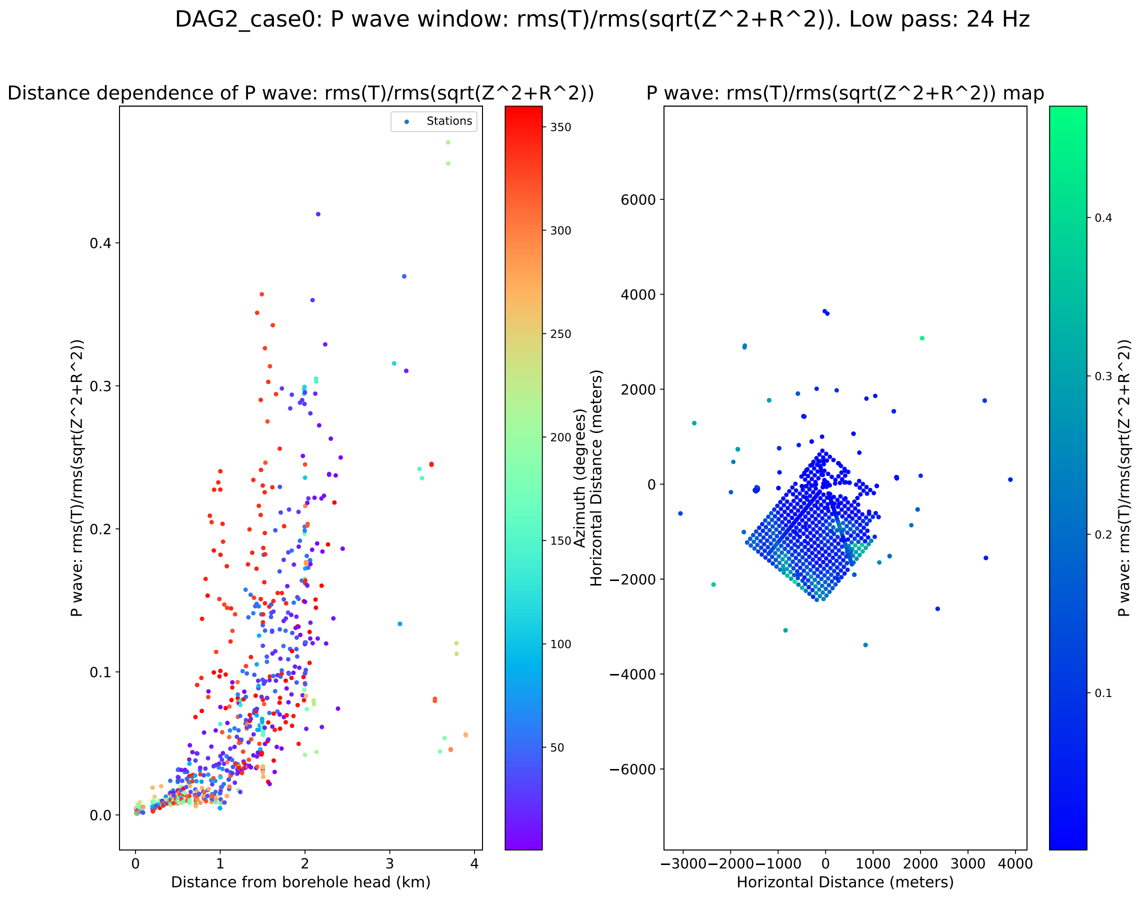


Figure 1.21: P wave component ratios of DAG-2 case 0. Both south southwestern and east southeastern sides of the site exhibit transverse component dominance, while other areas exhibit vertical and radial component dominance.

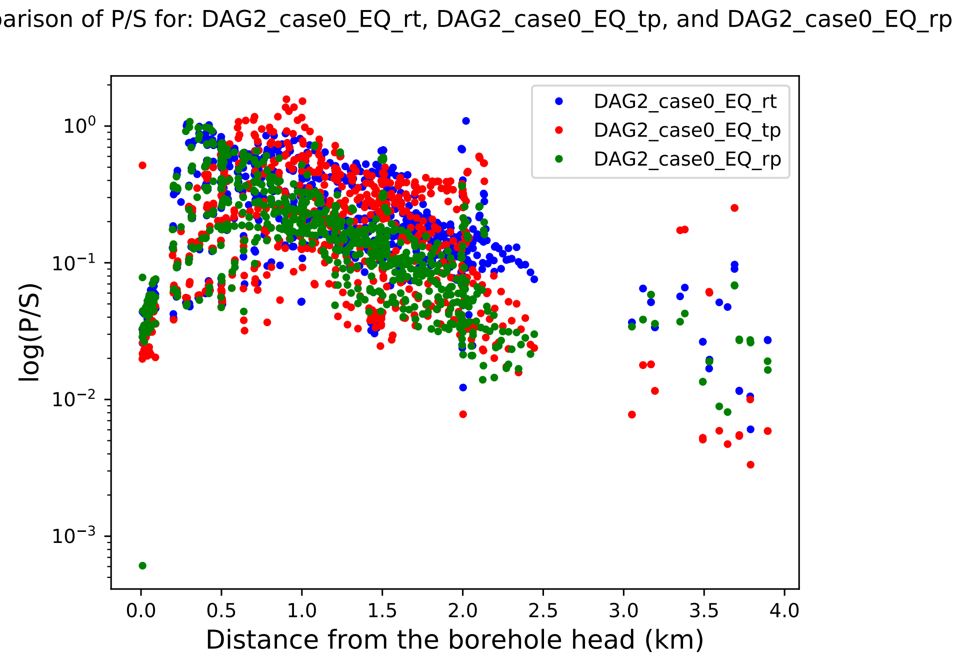


Figure 1.22: $\log(P/S)$ for three earthquakes of the same depth and magnitude, varying the off diagonal centriod moment tensor components: tp, rp, and rt. Even with such a minute difference, the earthquakes have different $\log(P/S)$ signatures.

1.5 Conclusion

Theoretically, P/S amplitude ratios can separate explosions from earthquakes (because shearing forces generate S-waves, where explosions do not [Houng (2018)]). One of the main difficulties in applying this technology to any particular setting is taking into account the surrounding geological setting. We tested the influence of the DAG Site subsurface structure on the ability to discriminate between explosions and earthquakes using the $\log(P/S)$ metrics, as well as a few others.

The major conclusion we draw from this study is an answer to the title of this report. Explosions look like earthquakes for large distances from the borehole due to complex wave interaction influenced by the 3D medium.

1.6 Future Work

Further investigation into the effect of distance on accurate source discrimination would be conducted. Specifically, the region between $2.5 - 4$ km is of particular interest to clarify the transition of $\log(P/S)$ behavior from near-field to far-field. These results would further illuminate the geological differences between the northern and southern regions of the DAG site. We could also determine for what distance from the borehole can we accurately distinguish between explosions and earthquakes.

Effects of mesh variation similar to the DAG-2 explosion cases would be of interest for earthquakes, as well as the effect depth on earthquake wave propagation.

Finally, variation of the number of geological layers is also a direction of further study in the field of seismic wave propagation through non-homogeneous media.

1.7 Acknowledgements

The authors would like to thank Ting Chen and Carene Larmat for their mentorship and continued patience, as well as Daniel Israel, Madison Andrews, Garry Maskaly, and Katrin Hammerling for making the XCP workshop possible virtually.

__J. Rubin Abrams__ is a rising 4th year graduate student at the University of Arizona, working on his PhD in Applied Mathematics studying integrable systems and soliton theory. He received his undergraduate degree majoring in Math and minoring in Computer Science from the University of Arizona. He is interested in becoming a working mathematician in industry or national lab working with scientific visionaries to tackle issues on a national level.

__Asher May__ received his B.S. Physics and B.S. Pure Mathematics from the University of New Mexico. He is interested in applications of mathematics and physics to climate and environmental modeling.

Bibliography

Alicja Caputa, Adam Talaga, and Łukasz Rudziński. Analysis of post-blasting source mechanisms of mining-induced seismic events in rudna copper mine, poland. *Contemporary Trends in Geoscience*, 4, 10 2015. doi: 10.1515/ctg-2015-0003.

History.com Editors. Nuclear test-ban treaty, nov 2009. URL `https://www.history.com/topics/cold-war/nuclear-test-ban-treaty`.

Soung Eil Houng. Discrimination between natural earthquakes and explosions based on the azimuthal distribution of the s/p amplitude ratios. *Bulletin of the Seismological Society of America*, 108(1):218–229, feb 2018.

Dimitri Komatitsch and Jean-Pierre Vilotte. The spectral element method: An efficient tool to simulate the seismic response of 2D and 3D geological structures. *Bulletin of the Seismological Society of America*, 88(2):368–392, 04 1998. ISSN 0037-1106.

Dimitri Komatitsch, S. Tsuboi, Chen Ji, and Jeroen Tromp. A 14.6 billion degrees of freedom, 5 teraflops, 2.5 terabyte earthquake simulation on the earth simulator. pages 4– 4, 12 2003. ISBN 1-58113-695-1. doi: 10.1109/SC.2003.10023.

Dimitri Komatitsch, Gordon Erlebacher, and David Michéa Dominik Göddeke. Higher-order finite-element seismic wave propagation modeling with mpi on a large gpu cluster. *Journal of Computational Physics*, 229:7692–7714, jun 2010.

Dimitri Komatitsch, Jean-Pierre Vilotte, and Jeroen Tromp. *SPECFEM3D Cartesian User Manual 3.0*. CNRS, Princeton University and ETH Zurich, https://github.com/geodynamics/specfem3d/, 3.0 edition, jul 2018.

Colin T. O'Rourke, G. Eli Baker, and Anne F. Sheehan. Using p/s amplitude ratios for seismic discrimination at local distances. *Bulletin of the Seismological Society of America*, 106(5): 2320–2331, oct 2016.

Catherine M. Snelson, Robert E. Abbott, Scott T. Broome, Robert J. Mellors, Howard J. Patton, Aviva J. Sussman, Margaret J. Townsend, and William R. Walter. Chemical explosion experiments to improve nuclear test monitoring. *Eos*, 94(27):237–239, jul 2013.

Wikipedia.com. Nuclear test-ban treaty, jul 2020. URL `https://en.wikipedia.org/wiki/Comprehensive_Nuclear-Test-Ban_Treaty`.

Chapter 2

Using RMI to Study Solid Media Under Extreme Strain Rates and Pressures Via Shock Loading

Team Members
Joshua Dyer and Crystal Ottoway

Mentors
Matthew Hudspeth and Michael Prime

Abstract

In this study, Richtmyer-Meshkov Instabilities in solids were simulated in FLAG to calibrate an elastic-perfectly plastic (EPP) strength model for copper at a drive pressure of 21.5 GPa by using jet length and maximum jet velocity as strength metrics. The resulting strengths are 405$\pm$200 MPa using maximum jet velocity and 365$\pm$30 MPa using jet length. Effects of viscosity in a tamper material were also investigated in terms of the maximum jet velocity, jet length, and shock wave perturbation decay, and the results are discussed considering viscosity values obtained in literature. Jet length and maximum jet velocity were found to be sensitive to viscosity while the shock perturbation was insensitive to viscosity. Multiple driver strength models are also used (EPP, elastic, hydrodynamic) to further investigate shock wave perturbation decay and to draw conclusions of assumptions made in simulation by published works. Driver strength models were found to have a significant effect on the shock perturbation.

2.1 Introduction

Richtmyer-Meshkov instability (RMI) in solids describes the growth of irregularities (jets) on the interface of shocked material with initial sinusoidal perturbations on the back surface. The shock can be created via flyer plate impact or high explosives. In the simulations performed during this study, the shock was created using a flyer plate impact. A common motivation behind RMI experiments in solids is to capture the strength of a material at an elevated strain-rate and elevated pressure to calibrate constitutive models for simulation (Prime, 2017).

Figure 2.1 shows the typical setup for solid RMI experiments using flyer plates. The impact of the flyer with the driver will create a right-going shock in the driver and a left-going shock in the flyer. When the right-going shock impinges upon the sine wave interface between the driver and tamper, the sine wave interface profile can invert and jets will form from the initial valleys. These jets are often referred to as spikes, but in this paper they are called jets. The jet forms as the shock wave reaches the valley of the sine wave before the peak, giving the valley a particle velocity while the peak is still stationary. This promotes the surrounding material to flow in the direction of the jet formation in a focusing manner. We define the jet length as the distance from the jet to the bubble, where the bubble is the region that was the peak of initial sine wave, as shown in Figure 2.1.

The jet formation is largely governed by the following four aspects: (1) constitutive behavior of the material, (2) drive pressure, (3) shape of the sine wave interface, and (4) Atwood number. The following sections will discuss these in more detail.

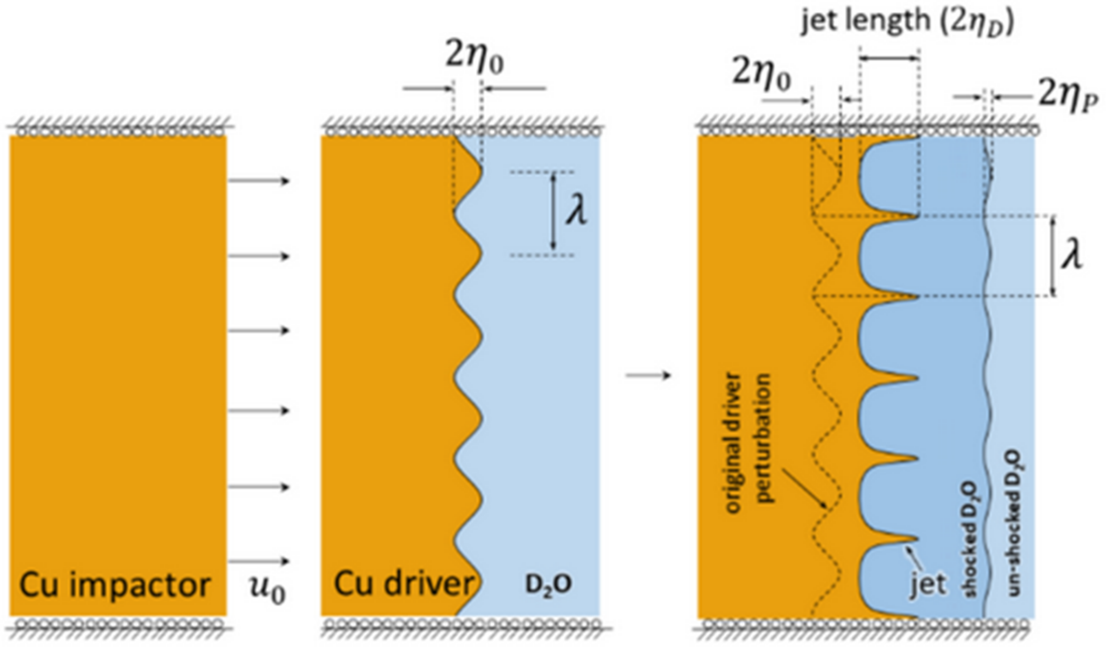


Figure 2.1: Setup of solid RMI flyer plate experiments with tamper.

2.1.1 Constitutive Behavior of Material

The constitutive behavior of a material refers to its strength and how it reacts under compressive loading conditions. In the current study, three different strength models have been implemented, namely hydrodynamic, elastic perfectly plastic (EPP), and purely elastic.

In order to assess the response of a material with negligible strength, we have modeled the driver material as hydrodynamic. Using such an assumption results in no deviatoric stresses being present in the material, allowing for hydrodynamic flow.

We have also modeled the driver material using the EPP constitutive response, resulting in a finite yield response. This model does not include strain hardening from the accumulation of dislocation defects in the plastic regime and instead perfectly yields. It has been shown that at a specific point in phase space, the EPP model adequately describes the maximum jet velocity behavior of the RMI inversion (Prime, 2017).

In order to assess the effect of a nearly rigid driver, a purely elastic model has been implemented, thereby allowing for an infinite yield point. The linear-elastic region continues to infinity, resulting in no plastic deformation. As such, the material will always restore its initial shape after stresses have been relieved.

Figure 2.2 shows all three of these strength models on a stress-strain curve, which has been adapted from (Forbes, 2015).

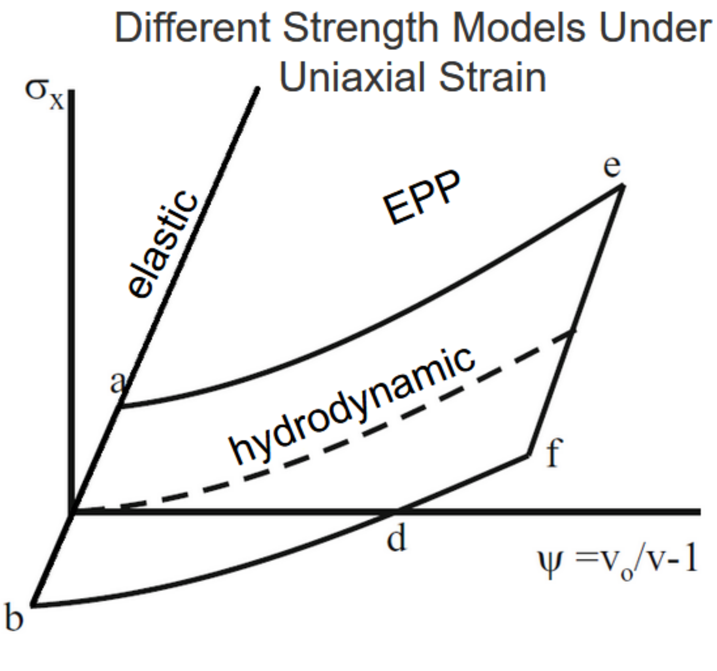


Figure 2.2: Different strength models shown on stress vs volumetric strain curve. (Forbes, 2015)

2.1.2 Drive Pressure

A simple description of a shock wave is a pressure front that causes a sharp increase in pressure in a very short time period and a very small distance, which is only a few free mean paths in thickness. This pressure created aft of the shock wave is called the drive pressure, and largely effects the jetting behavior of the driver. Higher drive pressures can result in longer jet profiles with higher particle velocities. As such, higher drive pressures result in jets that are more affected by tension and damage than lower drive pressures.

The drive pressure is dictated by the initial velocity of the flyer, as well as the material of the flyer and driver. To find the drive pressure, the P-u Hugoniot equations (Cooper, 2010) for the right-going shock in the driver (equation 2.1) and the left-going shock in the flyer (equation 2.2) are equated to provide continuity. Solving for particle velocity (u), drive pressure is then trivial.

$$P = \rho_0 C_0 u + \rho_0 s u^2 \tag{2.1}$$

$$P = \rho_0 C_0 (u_0 - u) + \rho_0 s (u_0 - u)^2 \tag{2.2}$$

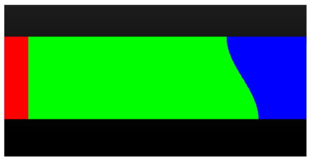
Where:
ρ_0 = initial density
C_0 = bulk sound speed (y-intercept of u-u plane)
s = slope of u-u plane
u_0 = initial flyer velocity

2.1.3 Sine Wave Interface Shape

The initial shape or amplitude of the sine wave interface between the driver and tamp materials will affect jet formation. The sine wave interface here is characterized by the dimensionless product of the wave number ($2\pi/\lambda$) and the initial amplitude (η_0).

$$\eta_0 k = \eta_0 (2\pi/\lambda) \tag{2.3}$$

Smaller values of $\eta_0 k$ will result in smaller jet formation amplitudes, while higher $\eta_0 k$ values will result in greater jet formation.This is demonstrated in Figure 2.3 and Figure 2.4 wherein the driver (green) is driven into the tamper (blue) via impact of the flyer (red).


(a) Initial sine wave interface. $\eta_0 k$ = 0.872

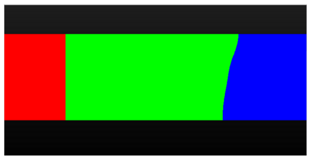

(b) Final jet formation. $\eta_0 k$ = 0.872

Figure 2.3: Jet formation given sine wave interface defined by $\eta_0 k$ = 0.872

Note that the value of $\eta_0 k$ defines the initial amplitude of the sine wave interface, which is represented in 2.3(a) and 2.4(a). Figures 2.3(b) and 2.4(b) demonstrate the jet formations after the shock interacts with the sine wave.

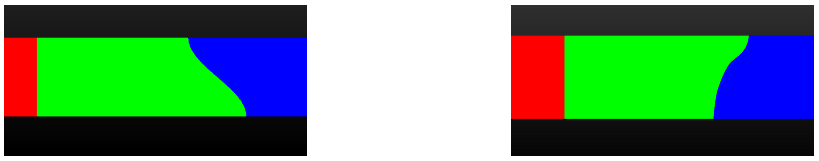

(a) Initial sine wave interface. $\eta_0 k$ = 1.74 (b) Final jet formation. $\eta_0 k$ = 1.74

Figure 2.4: Jet formation given sine wave interface defined by $\eta_0 k$ = 1.74

In summary, driver interfaces with higher values of $\eta_0 k$ have higher initial amplitudes and result in larger deformations and larger jet lengths.

2.1.4 Atwood Number

The Atwood number (A) is a dimensionless parameter that is used in the study of hydrodynamic instabilities. Generally, ρ_1 is the density of the of the driver media and ρ_2 is the density of the tamp media, before interaction of the shock-wave.

$$A = (\rho_2 - \rho_1)/(\rho_2 + \rho_1) \tag{2.4}$$

The Atwood number is important in characterizing RMI because it defines the asperity between driver and tamper densities, with smaller values of A resulting in more favorable jetting conditions. Additionally, it provides insight on the presence of damage. The presence of a sufficient tamper material will inherently reduce damage, as the jet will form in a compressed environment. Non-dimensional analysis suggests that an Atwood number greater than -0.9 is sufficient to remove significant effects of damage. It is also important to note that tamping can allow calibration of a material constitutive model at elevated driver pressures.

2.2 Methods

To simulate RMI experiments, we used FLAG, an arbitrary Lagrangian-Eulerian multi-physics code that includes staggered-grid and cell-centered hydrodynamics, radiation diffusion, magneto-hydrodynamics, and many material models (Rockefeller). Figure 2.5 shows the output of a coarse mesh used in FLAG simulation for visualization purposes; finer meshes were used for actual simulations.

The left region is the copper flyer, which has an instantiated velocity to create the impact and resulting shock wave The middle region is the copper driver, and the right region is the heavy water tamper. Sesame table #3336 was used for the flyer and the driver, and a Mie-Gruneisen EOS was used for the heavy water. Slideline boundary conditions were applied at material interfaces to allow sliding, and periodic boundary conditions were included so that only one half of the interface wavelength needed to be modeled. Tracers were arranged along the red lines that recorded time, location, and pressure to allow us to track the jet length, maximum jet velocity, and the shock wave perturbation in the tamper. Adequate depths for all three regions were used to eliminate shock reflections and to allow transition from coarse to fine mesh in the interface between the flyer and driver. A damage model was not included. As described in Section 4.1.1, the flyer and driver material are modeled with either EPP, elastic, or hydrodynamic strength models. Use of each model will be called out in the text below. The tamper material was modelled as either hydrodynamic, or with a constant viscosity model (ViscConst in Flag).

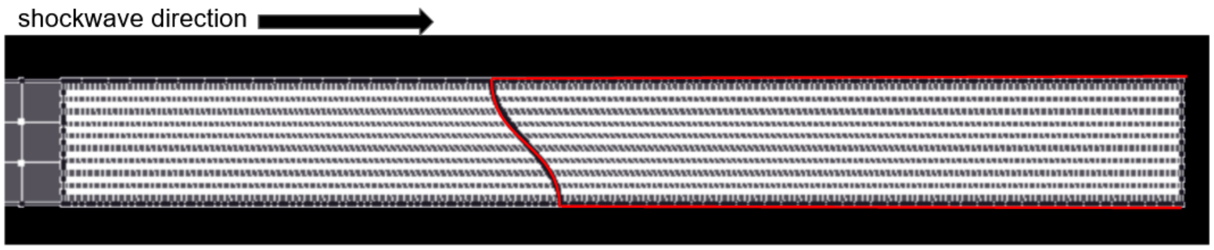


Figure 2.5: Mesh arrangement for FLAG simulations using a coarse mesh for visualization purposes

2.3 Calibration of EPP Strength Model

One of the main goals of the current RMI work is to capture the constitutive strength of a material at elevated strain-rates and pressures. The strength can be quantified using strength metrics, such as jet length or the newly-proposed metric of maximum jet velocity (Prime, 2017). Maximum jet velocity has the advantage that it is not appreciably affected by damage, as the maximum occurs early in the formation of the jet wherein tension is not yet present. However, obtaining the maximum jet velocity leads to larger uncertainties than the 1-2 percent accuracy that velocimetry is capable of in principle. Methods are currently being developed to lower the experimental uncertainties present in velocimetry. Using jet length as the strength metric has a large advantage in that it is more sensitive to strength leading to lower uncertainties. However, jet length measurements average over a much larger time interval that spans strain-rates differing in several orders of magnitude, which makes the resulting strength more applicable for validation purposes than strength model calibration. It should be also be noted that in an untamped environment, the resulting jet length is highly affected by material damage. Both strength metrics provide unique information, and thus both are investigated.

2.3.1 Mesh Convergence Study

Numerical convergence of the RMI mesh must be considered when working with RMI so as to achieve accurate simulation results when comparing to experimental data. Increasing the quality of the mesh and decreasing the cell size will increase maximum jet velocity and jet length. If an insufficient mesh is used, the strength estimate will be underestimated as shown by Prime (2017). Additionally, the Barton artificial viscosity model, which is being used in the current study, is more affected by the resulting mesh convergence when compared to the VonNeumann and Richtmyer (VNR) artificial viscosity model.

Figure 2.6 shows the effects of mesh quality on the maximum jet velocity. Each result was simulated using a flyer velocity of 1 km/s, a wavelength of 0.1 cm, $\eta_0 k$ of 1.0, an EPP strength of 900 MPa, a copper flyer and driver, and no tamper material (vacuum). Cell width was varied in relation to the wavelength (λ) for a range of appropriate cell widths including those close to a hypothetical zone size of zero. A cell width value of λ/100 was used for all following simulations for the driver and tamper regions as a balance between convergence behavior and computational cost.

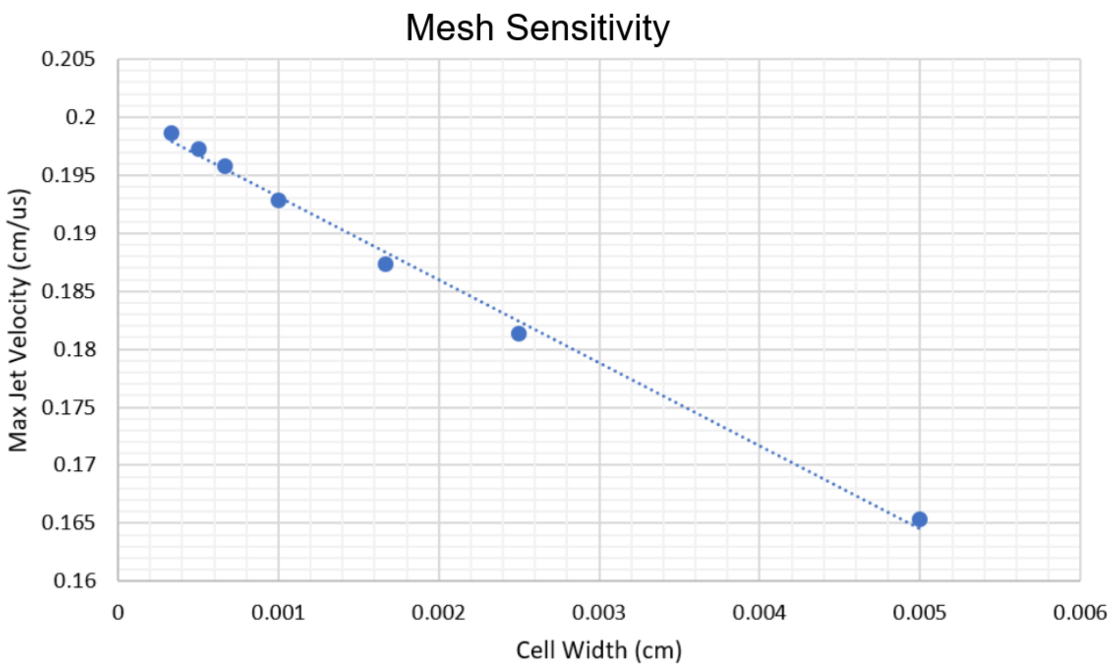


Figure 2.6: Mesh convergence study shows strong dependence on cell width regarding maximum jet velocity

2.3.2 Strength Calibration Results

To calibrate the strength of the EPP model, we used experimental data from Joe Olles (DCS19-1-033). Parameters for this data are as follows: OFHC copper flyer, OFHC copper driver, heavy water tamper, 1.023 km/s flyer velocity, 0.1 cm wavelength, and 0.97 $\eta_0 k$, resulting in a drive and release pressure of 21.5 GPa and 3.4 GPa, respectively. Figure 2.7 shows simulated spike and bubble velocities at different strength values for the same initial conditions as the experimental data which is represented as red dots. By looking at the zoomed portion of Figure 2.7, it can be seen that an EPP strength of 405 MPa strongly agrees with the maximum velocity of the experimental data. Since the uncertainty is largely conservative, the strength estimate from the maximum jet velocity strength metric becomes 405$\pm$200 MPa. The uncertainty quantification for this specific data set is rather large but will reduce with time as more effort is spent in post processing of PDV spectograms, thereby leading to a tighter strength estimate.

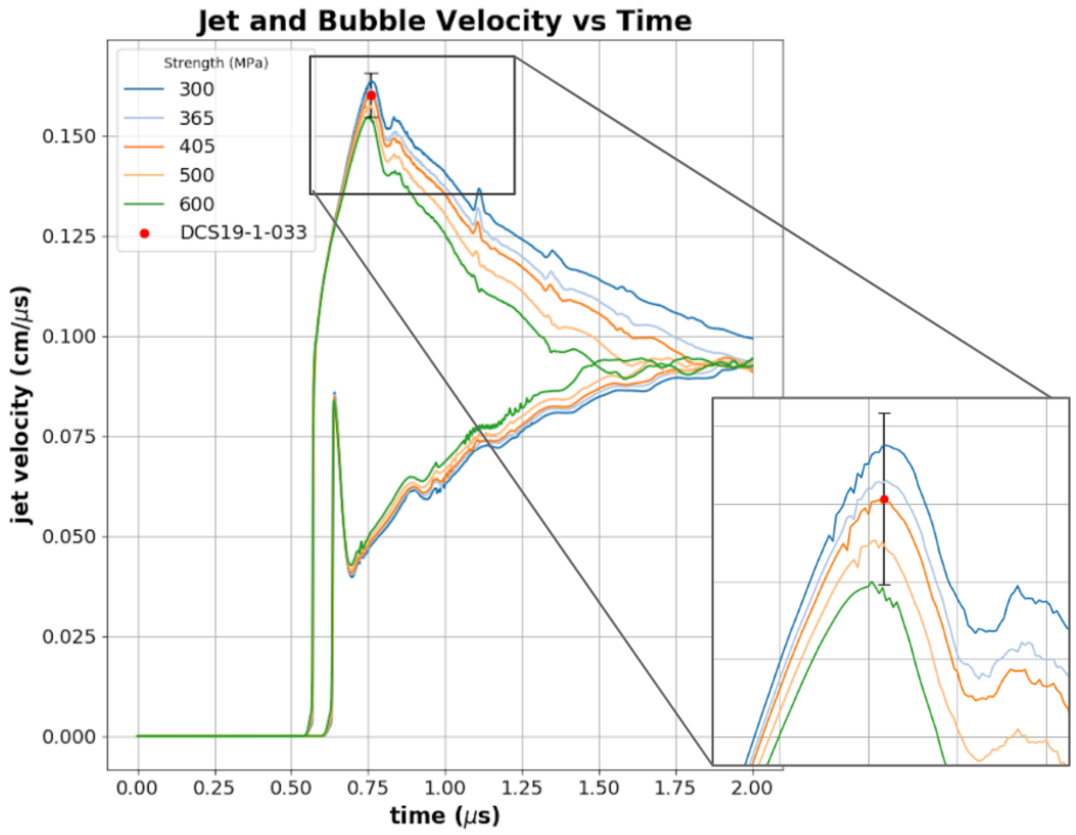


Figure 2.7: Spike and bubble velocities at varying EPP strengths

A similar approach was used when using jet length as the strength metric. Figure 2.8 demonstrates the jet lengths at different strength values, with the simulation parameters being exactly the same as in Figure 2.7. Comparing the uncertainties between the experimental data for jet length and maximum jet velocity, it can be seen that the jet length is more sensitive to strength with less uncertainty and larger differences between equivalent ranges of strengths. A strength of 365$\pm$30 MPa matches the experimental data closely.

The resulting strengths from both metrics are in general agreement with each other with only a 40 MPa difference between the two. Thus, there is no obvious evidence of damage in the jet for this specific experiment. Additionally, the tamped environment inherently reduces damage as the jet is in compression rather than tension, and an Atwood number of -0.78 indicates that significant effects of damage are not present.

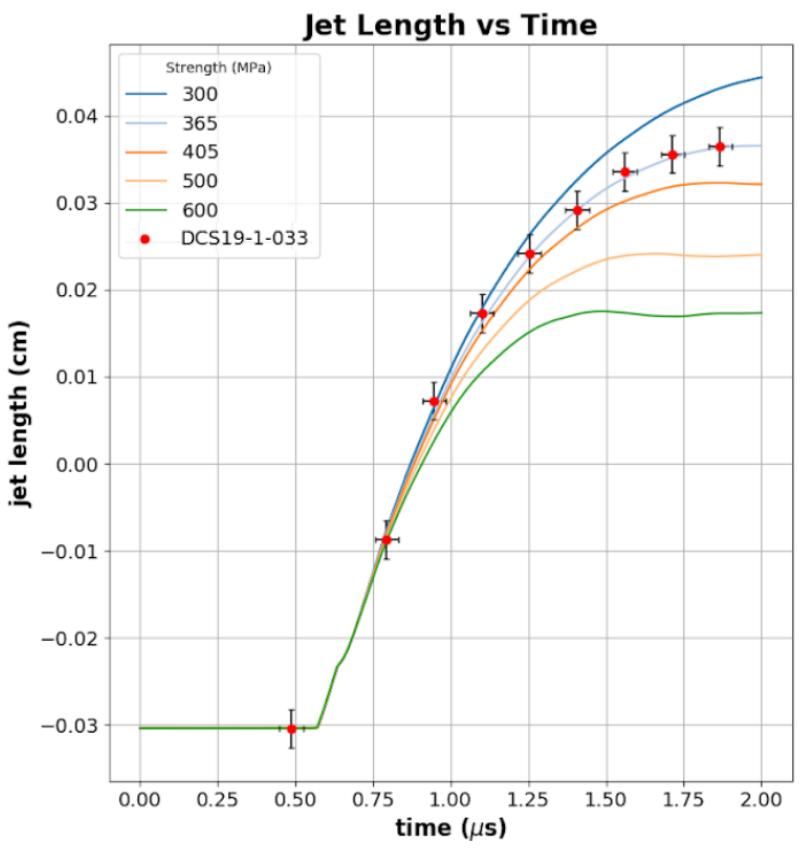


Figure 2.8: Jet lengths at varying EPP strengths

2.4 Tamper Viscosity Effects

Viscosity of the tamper material was investigated regarding its effects on jet length, maximum jet velocity, and shock perturbation decay. The motivation of this project was to emulate and verify viscosity values obtained by Sakharov et al. (1964) by using a similar techniques to the propagation of corrugations they used with more advanced simulation capabilities. The viscosity results from the Sakharov paper are several orders of magnitude greater than recent experimental results from the Abramson (2015) paper that use different techniques to find viscosity, causing need for investigation. The chosen scale for the simulated viscosity values covered the entire range of published viscosity values for water collected from the Abramson (2015) paper.

Sakharov et al. (1964) wanted to investigate a shock wave transferred into the tamp material through the sine wave interface of the driver. In their experiment, water was placed in the wedge-shaped container placed upon a driver interface, allowing the capture of the shock at multiple time steps. A shock is produced via an explosive charge that traveled through the copper interface and into the heavy water. The resulting evaluations of the perturbation decay from the recorded shock were used estimate viscosity values. An effort has been made in the current study to emulate the results obtained by Sakharov et al. (1964) in an effort to understand why the reported viscosity values were so large.

2.4.1 Maximum Jet Velocity and Jet Length

The results for the jet velocity and jet length were as expected given the change in viscosity. A range of 1-1,000,000 cP (Abramson, 2015) was used for the viscosity values. To give an idea of the scale of these units, atmospheric pressure water has a viscosity of 1 cP and honey has a viscosity of 10,000 cP. As the viscosity (μ) was increased, it was observed that the maximum jet velocity and jet length both decreased. Thus it can be concluded that viscosity acts as a damper for the maximum jet length and velocity.

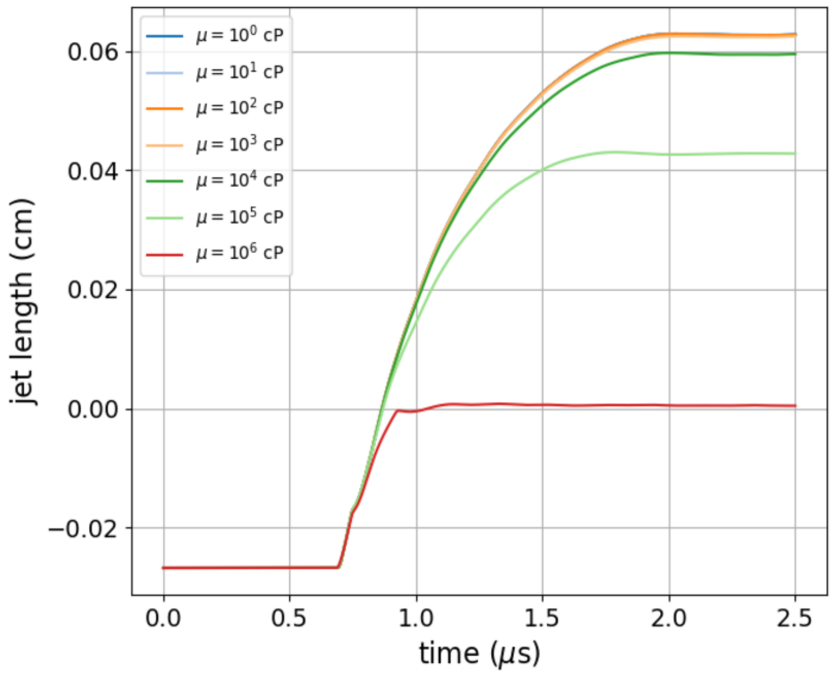


Figure 2.9: Maximum jet length at different viscosity values (μ) as a function of time

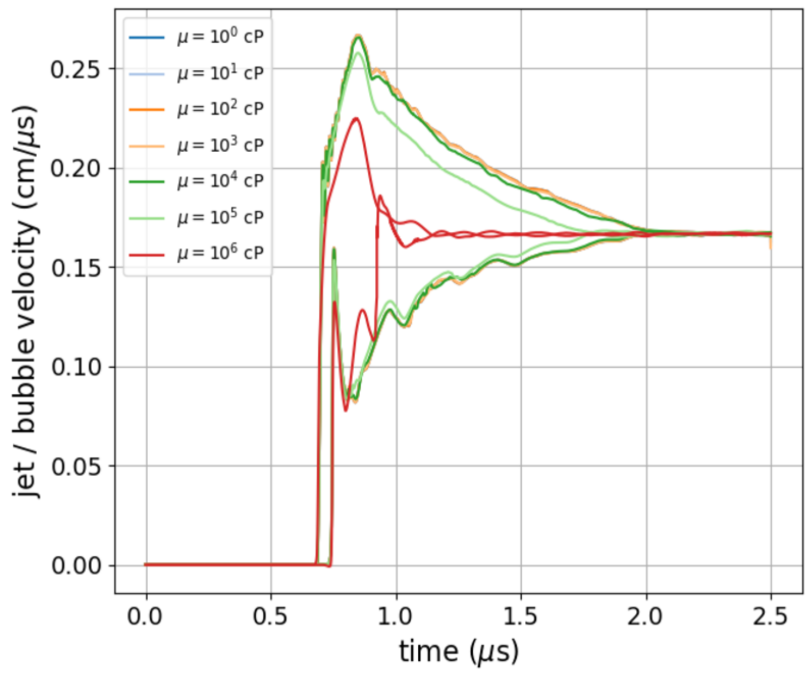


Figure 2.10: Maximum jet velocity at different viscosity values (μ) as a function of time

2.4.2 Shock Perturbation Decay

Due to the interface between the driver and the tamper, the shock wave in the tamper will have a sine wave profile instead of a planar profile. The amplitude of this shock wave will decay with time and follow the behavior of a damped second order system. We define the amplitude of the shock wave in the tamper as the shock perturbation. Figure 2.11 visually shows the shock perturbation in the tamper using a green driver and a tamper that is colored to represent pressure values.

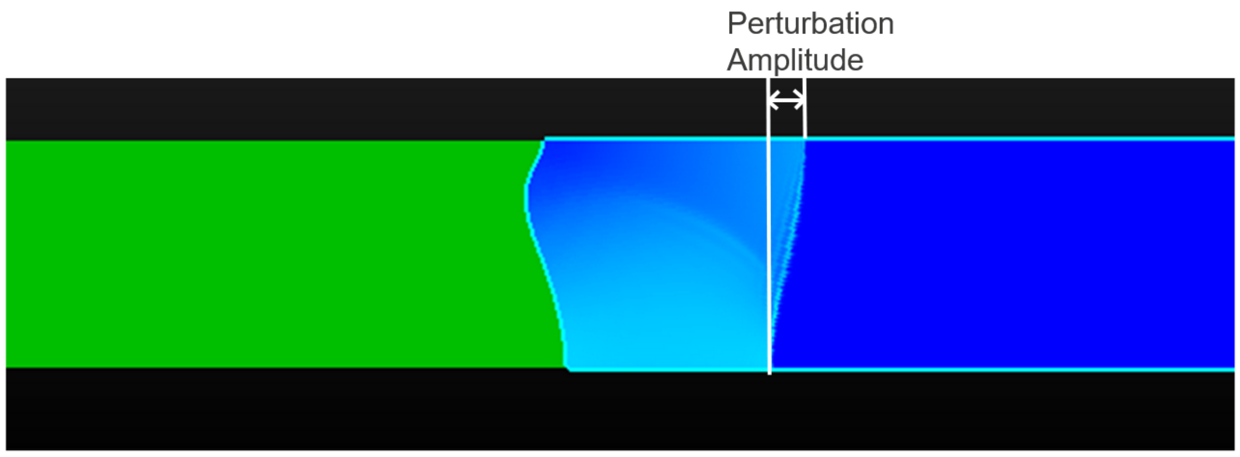


Figure 2.11: Visual representation of shock perturbation in tamper.

The results obtained for the shock perturbation decay were not as expected given the effect of viscosity on the resulting jet length and jet velocity, as described in Section 2.4.1. Figure 2.12 demonstrates the resulting perturbation decay obtained from the Flag simulations, suggesting that the shock perturbation evolution is insensitive to the viscosity of the tamped material. This was the case even when the viscosity for the heavy water was set to unrealistic values(10^5 Cp). More research on viscosity and its effects on perturbation decay is needed to understand this response. Additionally, verification of the viscosity model used in the simulation must be performed.

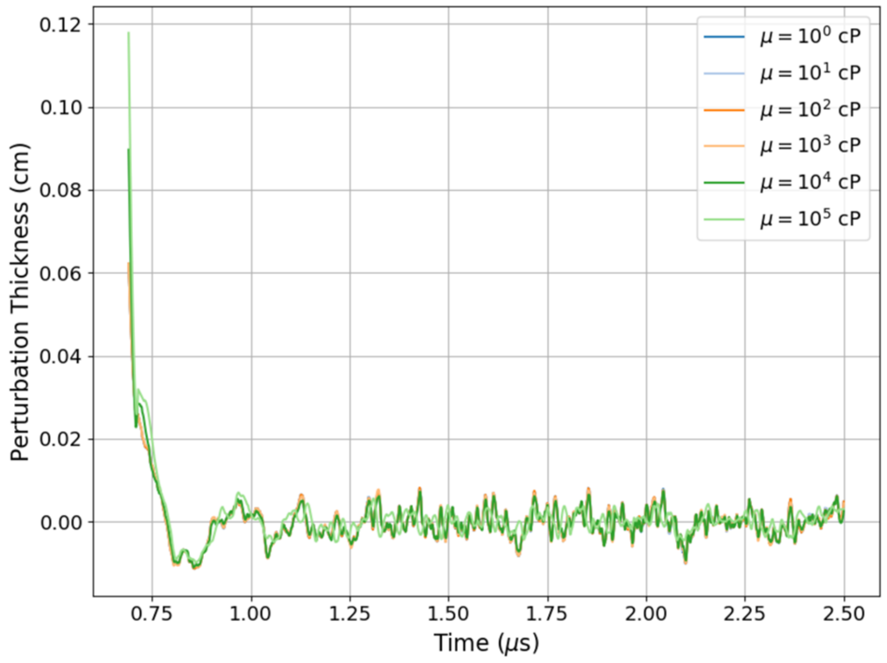


Figure 2.12: Shock perturbation amplitude at different viscosity values

2.5 Effects of Strength Model Variation

The effect of using different strength models was also evaluated as a method of emulating Sakharov et. al.'s results. Here we look at the perturbation decay in Sakharov's simulated and experimental result compared to the Flag outputs of different strength models. As described in section 5.1.1, the three selected strength models were EPP, elastic, and hydrodynamic. The EPP model will initially behave in an elastic fashion and then infinitely yield after passing the yield point. Unlike EPP, the elastic model only accounts for elastic deformations, and it does not include subsequent plastic deformations. The hydrodynamic model neglects strength in general and accounts for neither elastic or plastic deformations.

As demonstrated in Figure 2.13, the shock perturbation evolution is sensitive to the constitutive model selected for the driver. In Figure 2.13 the black and red graphs represent Sakharov's results while the rest are Flag outputs for different strength models.

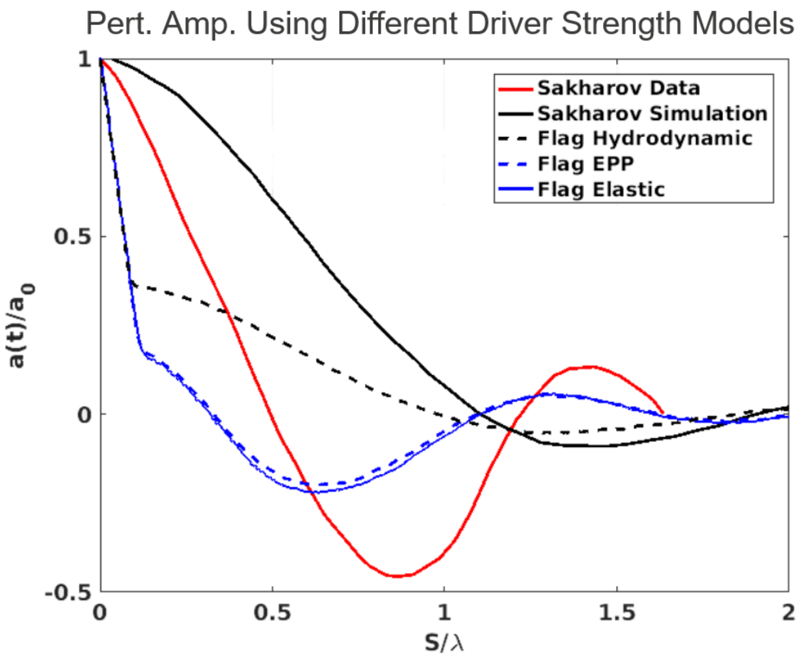


Figure 2.13: Perturbation amplitude found using different strength models

Looking at the decay characteristics in Figure 2.13 and time of inversion, which is the time the perturbation thickness first becomes zero for the solid and dashed graphs, we can suggest that Sakharov et. al. used a hydrodynamic model for the driver. Note however that this is never explicitly stated in Sakharov et. al.'s study; more research will be needed to find how Sakharov et. al. obtained their exact results.

2.6 Conclusion

There are several conclusions given the results of this project. Regarding the constitutive strength calibration for copper at a drive pressure of 21.5 GPa, the resulting strength was 405$\pm$200 MPa and 365$\pm$ 30 MPa from the maximum jet velocity and jet length strength metrics, respectively. For the variation of viscosity, the results concluded that the jet length and maximum jet velocity are sensitive to viscosity as it acts like a damper. This result was expected. However, the perturbation amplitude decay for the shock inside the tamp material is insensitive to viscosity. This was not expected given that jet length and jet velocity do change. Further study is needed in order to determine if this is indeed correct. When assessing the effect of driver strength on the resulting shock perturbation decay, the resulting outputs from Flag appear to suggest that Sakharov et. al. used a hydrodynamic driver in their analysis. The exact methods used by Sakharov et. al. to obtain experimental and simulated data need to be researched further as the Flag simulations performed during the project did not fully emulate their results for the perturbation decay in the liquid tamper.

2.7 Future Work

Future work must be performed to better uncover the initial conditions used in the Sakharov et al. (1964) study. The constant viscosity model implemented for the tamper also needs to be verified. An additional metric of interest would be to track the shock velocity in the tamper, as it would allow for comparison with modern velocimetry techniques. It is suggested that evolution of shock velocity in the perturbation would be more sensitive than perturbation thickness. Additionally, this elucidates the insensitivity of the perturbation thickness to the viscosity of the tamp material.

2.8 Acknowledgements

The authors would like to thank Matthew Hudspeth and Michael Prime for their guidance and mentorship throughout the course of the workshop. A special thank you would also like to be given to all organizers of the workshop for their hard work in setting up the workshop in such difficult and unpredictable times, as well as to those who gave lectures and to those who helped with computer issues. We students are very grateful for the opportunity to be involved in research at LANL, and this past summer was a great experience despite issues associated with a fully-online experience.

__Joshua W. Dyer__ is a senior at New Mexico State University pursuing a dual major in mechanical and aerospace engineering. He is the Vice President of the Society of Automotive Engineers Baja team at NMSU where he has previously served as the driveline subgroup lead, and he is also one of the drivers for competition. Following graduation in Spring 2021, Josh is interested in pursuing a masters degree in materials science.

__Crystal F. Ottoway__ is a recent graduate from Arizona State University with a B.S. in physics. She researched the electronic properties of solids using density functional theory as an undergraduate. Following a gap year from school, Crystal plans to start working towards a PhD in physics.

Bibliography

E.H. Abramson. Speculation on measurements of the viscosity of shocked fluid water. *Shock Waves*, (25):103–106, 2015.

Paul W. Cooper. *Explosives Engineering*. John Wiley, 2010.

Jerry W. Forbes. Shock wave compression in condensed matter. *Springer-Verlag Berlin*, 2015.

Mike Prime. Estimation of metal strength at very high rates using free-surface richtmyer–meshkov instabilities. *Journal of Dynamic Behavior of Materials.*, (3):189–202, 2017.

Gabriel M. Rockefeller. Lagrangian applications project public space. URL `https://xcp-confluence.lanl.gov/display/LAPP/Lagrangian+Applications+Project+public+space`.

A.D. Sakharov, R.M. Zaidel, V.N. Mineev, and A.G. Oleinik. Experimental investigation of the stability of shock waves and the mechanical properties of substances and high pressures and temperatures. *Doklady Akademii Naulk SSSR*, (159:5):1019–1022, 1964.

Chapter 3

Numerical Investigation of Explosive Particle Jetting

Team Members
Calvin J. Young

Mentors
Jonathan D. Regele, Yash Mehta

Abstract

A blast wave traveling through a region of particulate matter has been observed to produce distinct clusters and jets of particles expanding with the flow. This phenomenon has yet to be fully explained, and is difficult to investigate experimentally. Simulations will often use models for events happening at the particle scale, however these models are often based on assumptions from theory or empirical evidence. As such particle interaction models may be improved upon by further direct numerical investigation. A series of 2D simulations using the adaptive wavelet compressible flow code AWESUMM are performed in order to investigate this phenomenon qualitatively. Particles in this code are modeled as fully resolved cylinders via a volume penalization method. Phase interactions are captured by two-way particle-gas coupling and particle-particle collisions and momentum transfer. In this set of simulations, an incident planar shock is passed through a particle field, and the resulting flow field is allowed to evolve. Particle fields are varied in initial distribution in area fraction along the height of the domain. Particle trajectories and field area fractions are used to characterize evolution of the system over time. Further work will serve to shed more light on the mechanisms of jetting and validate particle models in use with other applications.

3.1 Introduction

Experiments investigating explosive particle dispersal are generally conducted by impulsively accelerating a dense field of particles with a shock or blast wave, resulting in an expanding particle cloud. Jet-like structures are commonly observed extending ahead of the bulk region of the expanding cloud; this phenomenon is termed 'Explosive Particle Jetting'. While this phenomenon has been widely observed on the macro scale, the mesoscale mechanisms by which it occurs is not fully understood. Direct experimental investigation of particle interactions is difficult, due to both the relatively small temporal and spatial scales on which these events are occurring. Such an experiment would also be prohibitively costly, due to the facilities, diagnostics, and expertise required to execute them using explosives.

A possible alternative is to simulate these experiments in order to resolve the fluid dynamics and particle interactions in order to better understand what is occurring. Fullscale simulations of explosive particle dispersal would however be unfeasible due to the relatively high computational cost of resolving across the wide range of spatial and temporal scales involved in the problem. Commonly, macro scale simulations employ models for particle scale interactions. To couple the momentum of the fluid and particle phases the particles modelled as points (Euler-Lagrange) or as a fluid (Euler-Euler). These models, based on theory or empirical observation, do not account for all of the relevant physics involved in the problem.

Mesoscale simulations of a simplified problem at the particle scale are able to shed some insight onto the mechanisms of particle jetting and clustering. A simulation involving tens to hundreds of fully-resolved particles can resolve phenomena occurring at much smaller scales than a simulation of a full macro-scale system. By directly capturing particle interactions and flow features, valuable information is gained that allows for the improvement and validation of models.

This study is performed to investigate the effect of initial particle distribution and perturbations in initial distribution on the development of the particle jetting phenomenon. In order to investigate this, several initial distributions of particles were designed for use as initial conditions in a simulation where they would be impulsively accelerated by a shock, and the resulting flow allowed to develop. The goal is to both re-create the jetting phenomena in a simplified problem via initial distribution, and identify possible mechanisms and potentiators of the jetting behavior.

3.2 Methods

3.2.1 Code Setup

Simulations of particle-curtain development presented in this study were performed utilizing the compressible flow code AWESUMM. The code solves elliptic and time-evolving equations via adaptive wavelet transforms with a prescribed computational error threshold. The fluid phase of the flow field is handled by the compressible Navier-Stokes equations. The particles were modelled as cylinders via an immersed boundary condition (volume penalization). Collision and momentum transfer between particles is handled with a hard-sphere collision model borrowed from molecular dynamics. To reduce computational costs the simulations were performed in 2D.

The Adaptive Wavelet Collocation Methods (AWCM) used in the code allows for DNS-level simulations to be performed to a high degree of numerical accuracy, along with a high degree of data compression. This is due to the ability of the code to perform adaptive mesh refinement based on adaptive-wavelet decomposition; wavelet modes at any point in the grid found to be extraneous are discarded, and the mesh accordingly updated. Combined with volume-based penalization for objects in the flow, this allows for the dynamic resolution of surfaces, to include moving obstacles.

The volume-based penalization scheme in use appends penalization terms handling the obstacle via the use of a transient masking function, whereby physical terms are removed inside the obstacle, and nonphysical viscosity is introduced. This allows continuity at the boundary between the flow and object, granting numerical stability. Further reading on wavelet and volume-penalization methods can be found in Schneider and Vasilyev (2010) and Brown-Dymkoski et al. (2013), respectively.

3.2.2 Initial Conditions

Initial conditions of the gas-phase are held constant across all cases, in order to provide an equal basis of comparison for evolution of the variations in initial particle seeding. Particles are modelled as cylinders placed into a two dimensional channel. A sub-region of the domain was chosen as the location for the initial placement of the particle cloud. In the initially defined particle cloud, the mean area fraction was held constant across all cases investigated, while the distribution of area fraction was varied. Area fraction being defined by the area occupied by particles as a fraction of the total area of the cloud at initial time.

The boundaries in the Y-direction are defined as periodic, allowing particles to exit and re-enter the domain, emulating the behavior of a wider domain. An incident planar shock, travelling in the positive x-direction, is initialized to the left of the particle curtain.

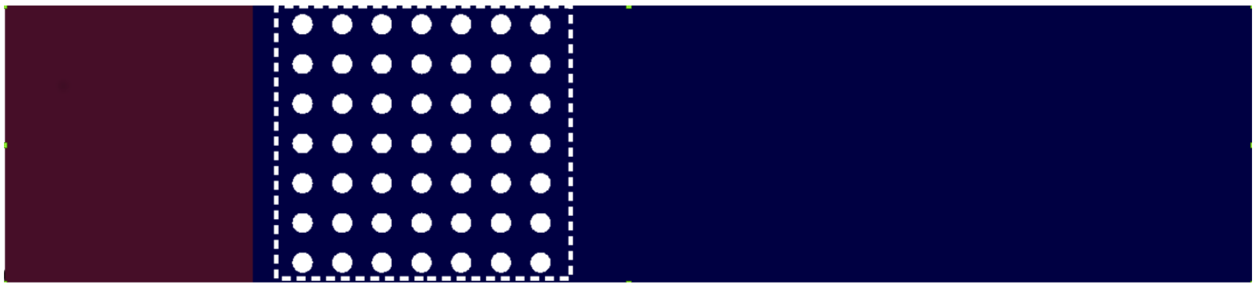


Figure 3.1: Initialized domain. Note the initial particle cloud outlined, location of shock front at initial time.

The strength of the incident shock is set to a Mach number of 1.67. The Reynold's number for the particles was chosen to be 300. Particle density is set to be five times the post-shock density of fluid following the incident shock. A low particle density is chosen in order to encourage quicker entrainment of particles and rapid development of the flow. The governing equations of the code are non-dimensional. Characteristic length is defined in terms of particle diameter D_p. The physical meaning of particle diameter is related by the chosen Reynolds number, which is defined by the post-shock conditions of the gas.

3.2.3 Particle Seeds

To study the effects of initial distribution of area fraction on the formation of jetting behavior, four initial configurations of particles in the curtain were chosen. The configurations were chosen in order to study the effects of both orderly and random initial distributions, as well as the development of an imposed perturbation. It is thought that in doing so, certain behaviors and mechanisms could be isolated and identified. In each case the mean area fraction of particles in the curtain was defined as 0.2. For Case 1, Fig.3.2a, particles were seeded on an evenly spaced-square grid, yielding a uniform distribution of area fraction. In Case2, Fig.3.2b, particles were placed at in the cloud following a uniform random distribution, allowing random fluctuations in area fraction. This configuration emulates a distribution of particles analogously to experiments such as Wagner (2012), although in a much simplified manner.

A sinusoidal variation in area fraction is imposed in Cases 3 and 4, in order to explore a the effects of a high-amplitude initial perturbation on development of the accelerated particle cloud. A sinusoidal perturbation does not happen in nature, but can be used to represent some arbitrary local perturbation. Such a perturbation has been hypothesized to be a potentiaor of jetting behavior. The sinusoidal variation is imposed by dividing the domain inside the curtain by horizontal bins in the Y-direction, and applying a sinusoidally weighted volume fraction to each bin, by which the number of particles per bin would be defined. This weighting can be seen clearly in Fig.3.2c. Case 4, Fig.3.2d, is a variation on Case 3, where a bounded, random shift in location has been applied to each particle's initial position. The random shift in position is applied in order to reduce the effects of the even, regular structure in Case3, as to provide a more realistic initial condition more analogous to what might be seen in a an experimental setting.

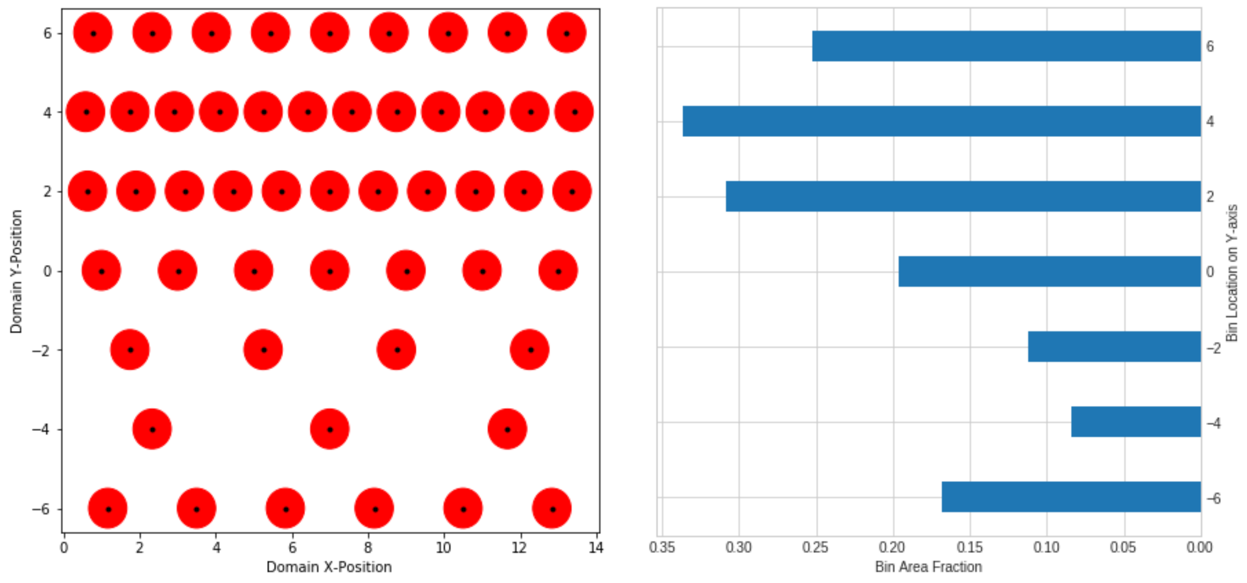
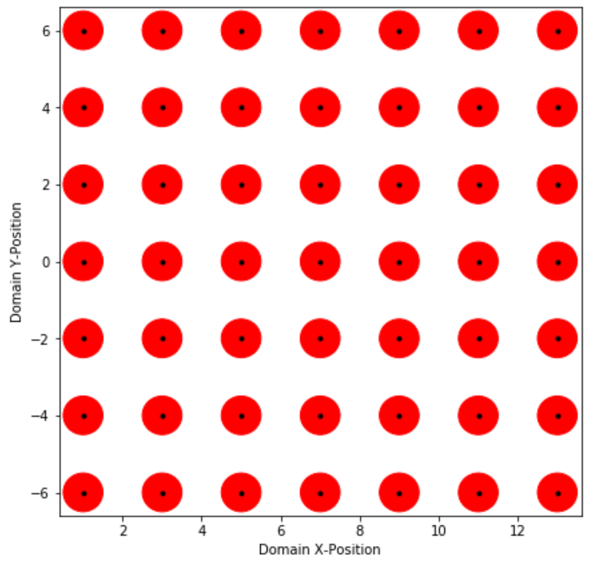


Figure 3.3: Volume Fraction Distribution of Sinusoidal Case

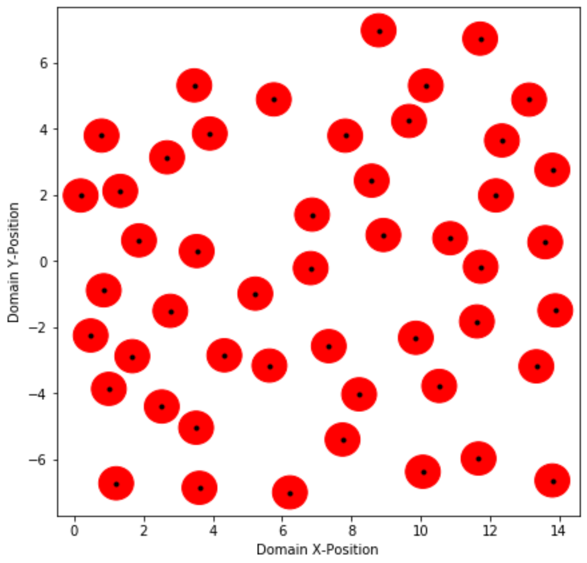
3.3 Results

3.3.1 Curtain Evolution

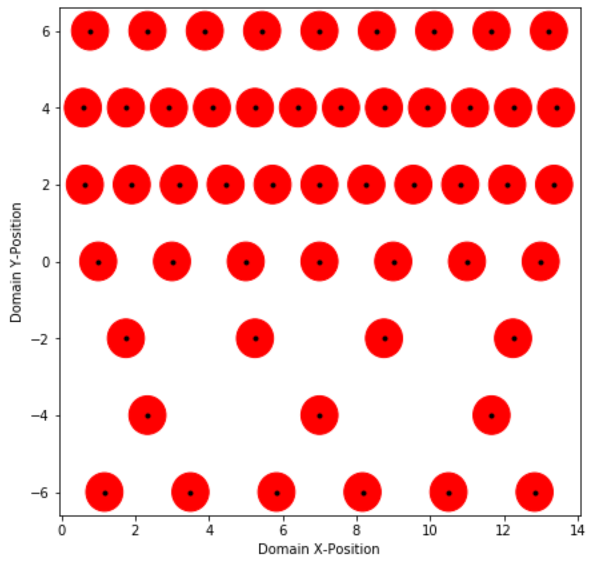
Each of four cases of initial condition was simulated for a time of 50 τ, where τ is non-dimensional time, allowing the particle cloud to travel the entire length of the domain. In order to visualize the



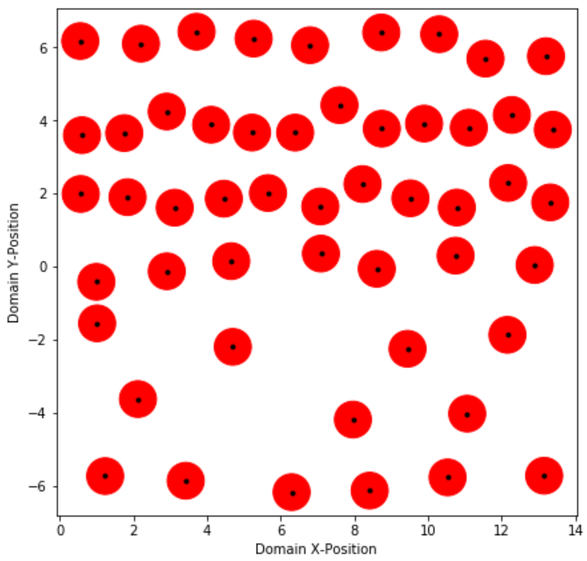
(a) Case 1



(b) Case 2



(c) Case 3



(d) Case 4

Figure 3.2: Initial particle distributions in curtain.

evolution of the particle curtain in each case, psuedocolor plots of fluid momentum in the X-direction overlaid with particle position were created (Figs. 3.4 - 3.7). These figures are comprised of plots of the entire computational domain, starting with time 0 and stepping every 10τ to time 50. As particle entrainment is one of the hypothesized mechanisms of jetting, the quantity of X-momentum is fitting for visualization of that phenomenon.

The initial seeding conditions of Case 1 are artificial, giving insight into how a highly structured regular cloud will evolve. Due to the uniformly ordered structure, collisions between particles were predicted to be almost one-dimensional In Case 1, Fig.3.4, the shock passage is observed to compress the particle cloud, until late time when particles to the right of the cloud begin to expand. Symmetry is maintained throughout, until late time when particles at the left side of the curtain to scatter due to some instability. Between rows of particles channeling behavior is observed, where the fluid-phase develops high momentum in a nozzling effect. While no jetting behavior was observed, insight into the behavior of the cloud as it is compressed was gained.

The uniform random distribution of Case 2, Fig.3.5, exhibited much of the same behavior as in Case 1. Bulk cloud compaction is observed, along with formation and breakup of small clusters of particles. fluid exhibits channeling between particles. Significant motion of particles in the y-direction is observed, indicating significant entrainment with the fluid phase and momentum transfer between particles. Channeling of the fluid phase is also observed between particles. Here no jetting behavior is evident, however towards the top of domain in the region of initially low area fraction, high fluid velocity relative to the high area fraction regions is observed.

Cases of initial distribution with a sinusoidal perturbation in area fraction exhibited behavior suggesting a significant effect of perturbation in area fraction. In Case3, Fig.3.6,prominent channeling of fluid phase in initially low area fraction regions and conversely low fluid momentum in initially high volume fraction regions. Strong fluid channeling effects were also exhibited in low area fraction regions, as well as entrainment. Significant entrainment of particles into regions initially low area fraction area indicates that an imposed initial perturbation has a strong effect on the evolution of the flow field

Case 4, Fig.3.7 exhibited behavior of possible early stages of jetting. The channeling effect is not as pronounced as in the previous case, due to the position scattering destroying the regular order of Case 3. The random nature however does look to impart greater fluid momentum and particle movement in the y-direction. Heavier entrainment of particles into the initially low volume fraction region than in the previous case may be due to this re-direction of momentum. At late time, this case shows evidence of early-stage jetting at late time. Towards bottom of the domain, particles are seen to concentrate in the region of initially low volume fraction, and a region of strong fluid momentum in the x-direction is observed extending through and ahead of that region. This case shows the strongest evidence of jetting so far in this study.

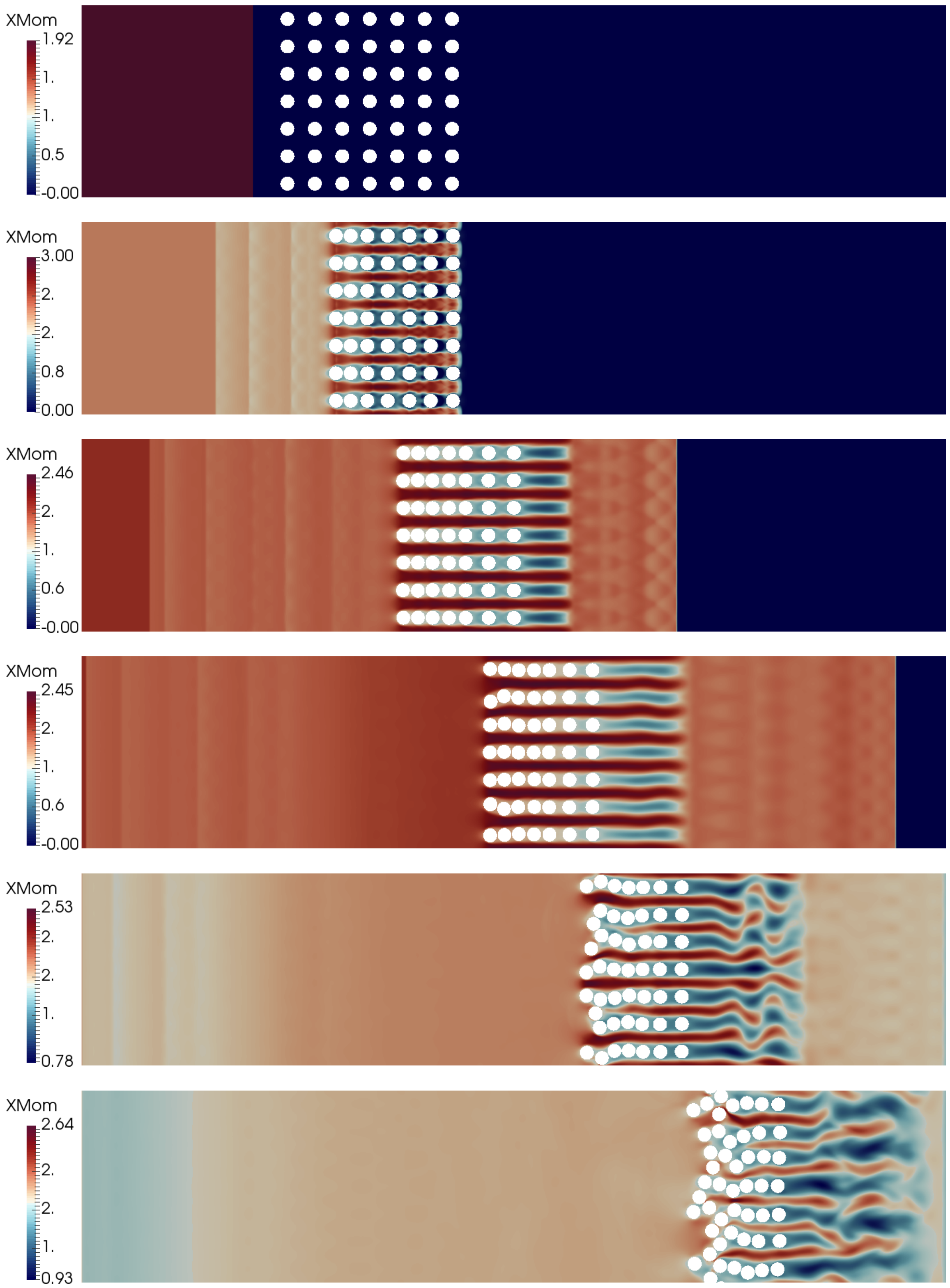


Figure 3.4: Magnitude of momentum in the X-direction, Case 1

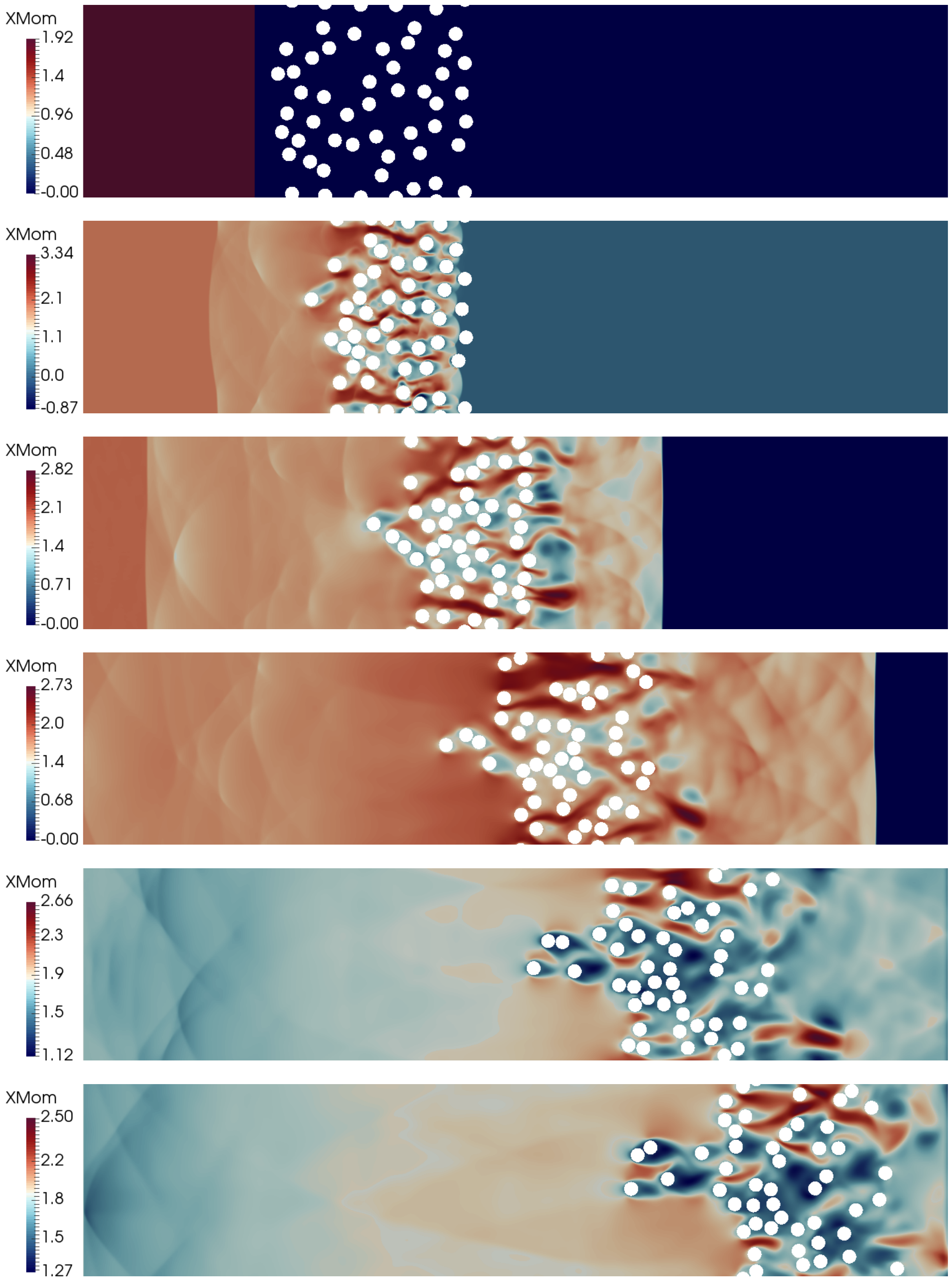


Figure 3.5: Magnitude of momentum in the X-direction, Case 2

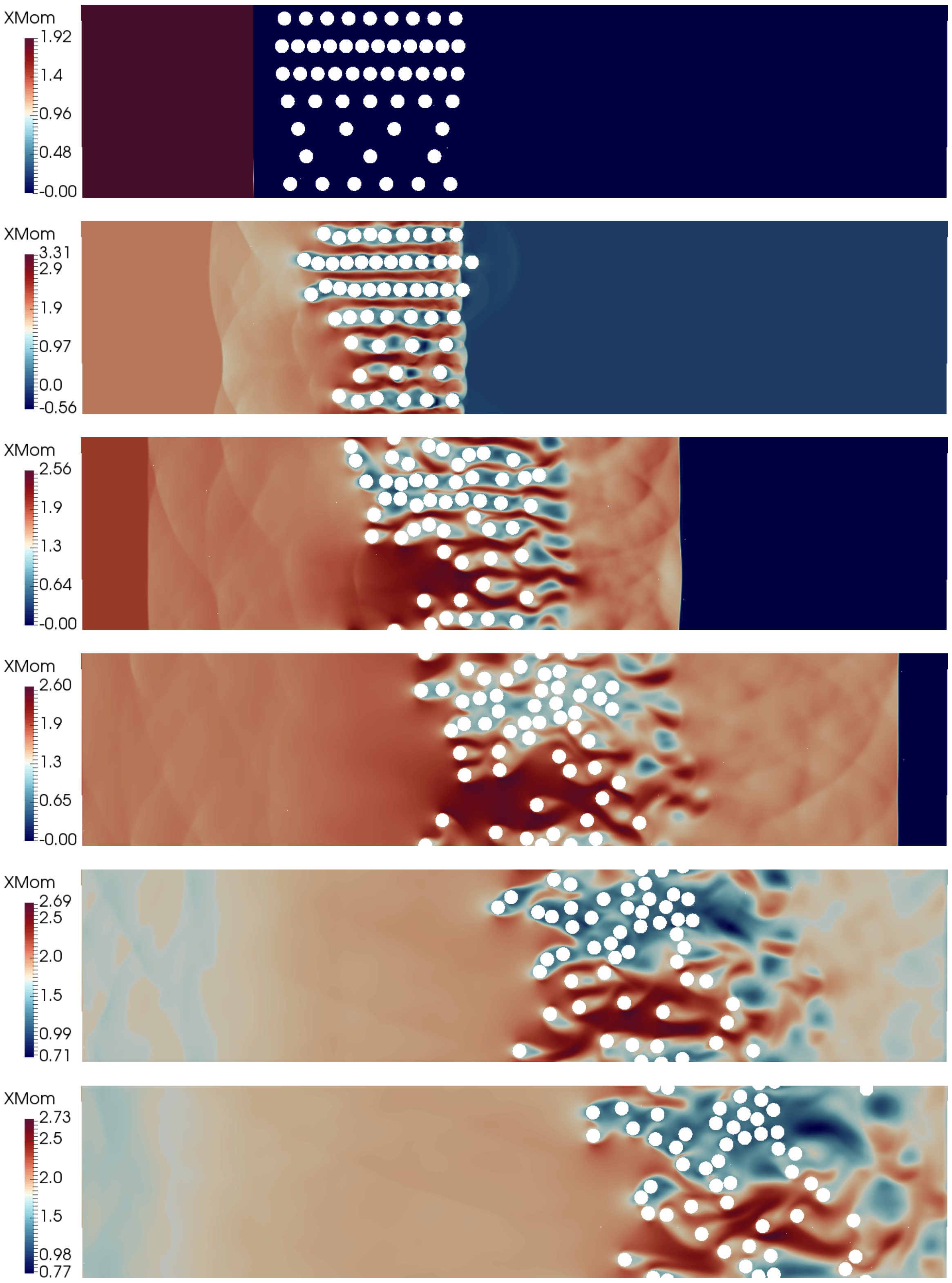


Figure 3.6: Magnitude of momentum in the X-direction, Case 3

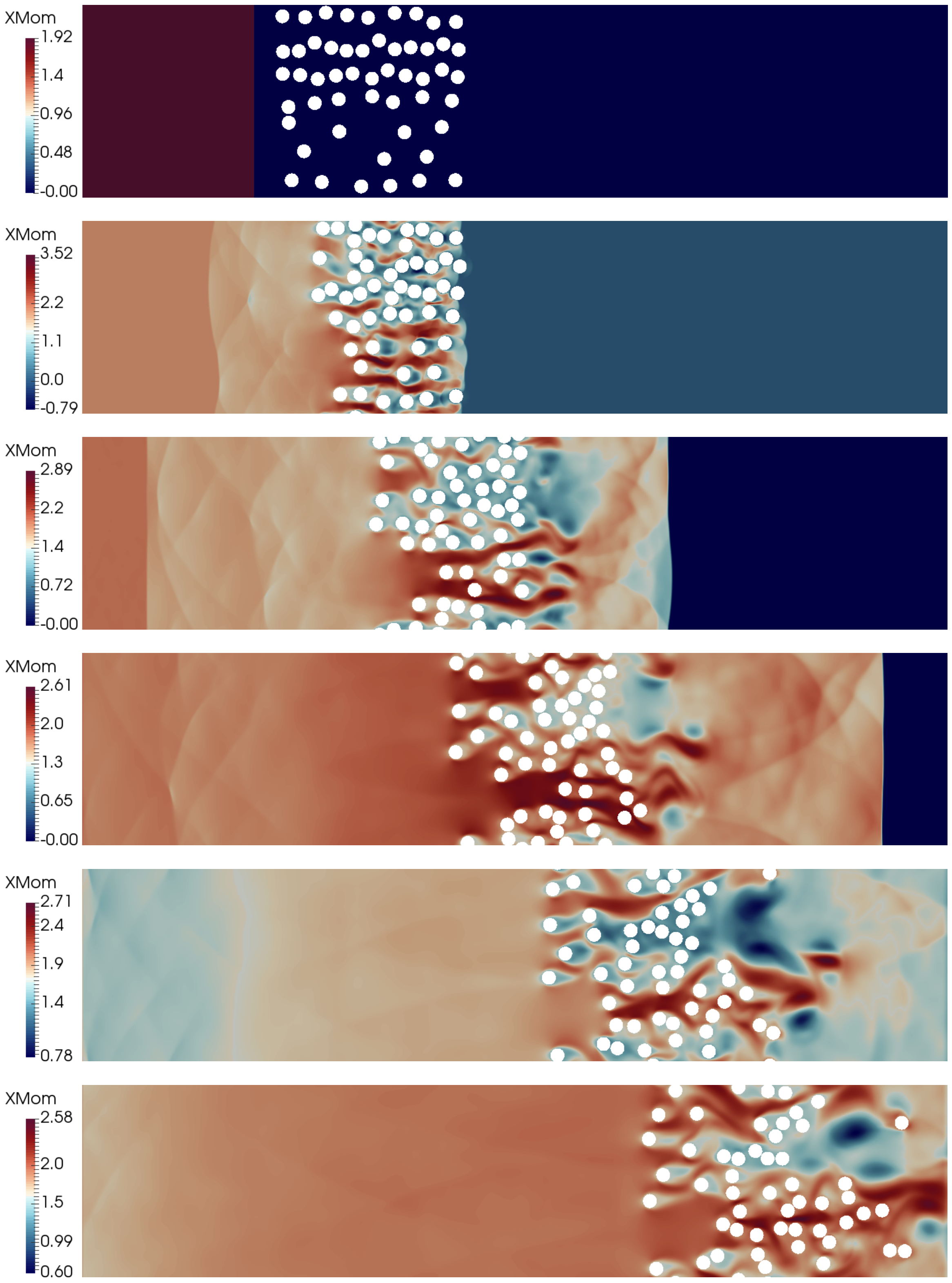
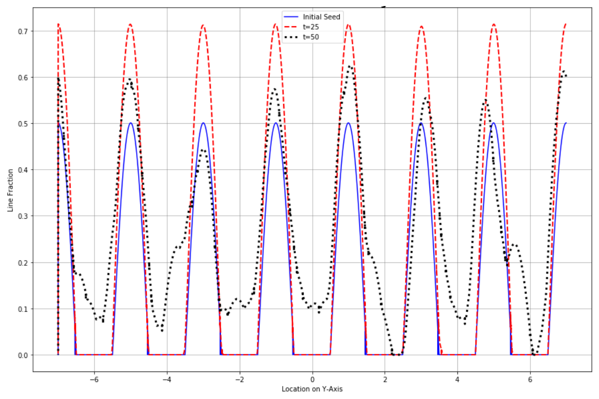
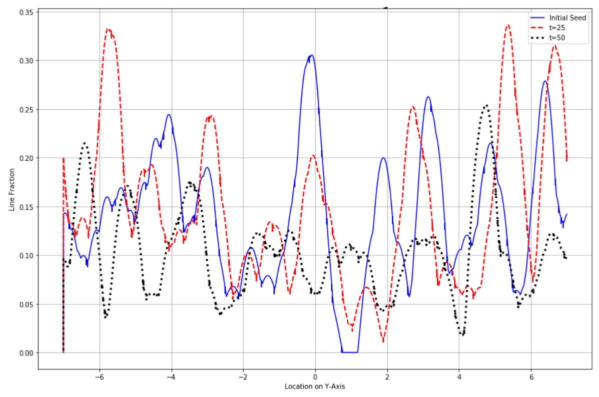


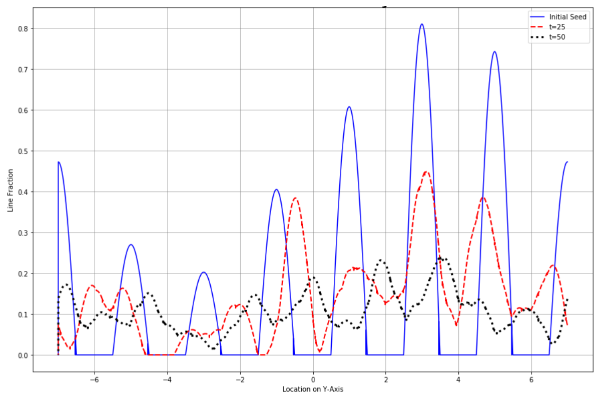
Figure 3.7: Magnitude of momentum in the X-direction, Case 4



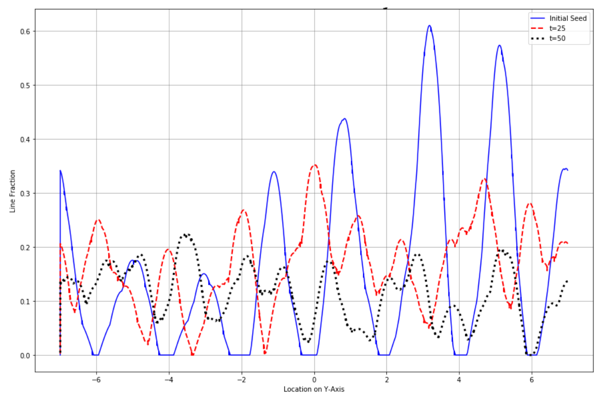
(a) Case 1



(b) Case 2



(c) Case 3



(d) Case 4

Figure 3.8: Evolution of line fraction for each initial distribution case.

3.3.2 Area Fraction Evolution

Line fractions are a one-dimensional analogue to area fraction. This value describes the fraction of the sub-domain of the particle cloud inhabited by particles, viewed in the transverse direction from a position on the of viewing. As this study imposes an artificial perturbation in area fraction along the y-direction, line fractions are taken from the y-axis in the x-direction. This provides a quantitative analysis of the redistribution of area fraction as the flow evolves.

Case1 saw compaction of the particle cloud, maintenance of symmetry, and little movement in y-direction. This behavior is illustrated in Fig.3.8a, where the linear compaction of the cloud decreased the cloud width, causing a spike in area fraction. At late times the cloud began to expand and scattering began to occur, reducing area fraction. In Case2, Fig.3.8b,no initial structures were observed. As the flow developed, clusters formed and dispersed, but there was no evidence of the development of coherent structures. It is worthy to note that the initial random distribution evolves into a more even distribution across the cloud, indicating movement of particles from areas with initial high amplitude spikes in area fraction to areas of low area fraction.

The cases of initial sinusoidal perturbation in area fraction displayed similar behavior.The initial perturbation in distribution spreads into a more uniform distribution with lower amplitude variations in area fraction. The regularly ordered Case 3, Fig.3.8c, shows evidence of the initial structure being maintained through late-time. Case4, Fig.3.8d, shows no evidence of retention of the initial structure of the particle cloud. This verifies that particles are being entrained from regions of initially high to low volume fraction. Such a redistribution shows strong transverse motion, suggesting that jetting is occurring.

3.4 Conclusion

In this project simulations were performed at the mesoscale to direct particle behavior in a densely laden field. Particle clouds of various initial distribution were impacted with a planar shock wave in order to study the relation between area fraction variation and the occurrence of jetting. Clouds of uniform gridded and random distributions in area fraction did not exhibit obvious jetting behavior. The two cases of sinusoidal distribution in area fraction evolved clustering and entrainment in a manner suggesting that an initial perturbation promotes jetting. Due to the length of the domain in use, these simulations must be repeated with an extended domain in order to allow the flow to evolve further. Future work studying imperfect collisions has been identified as another route of progress, as energy loss due to collisions has been hypothesized to promote clustering.

3.5 Acknowledgements

I would like to thank Jonathan Regele and Yash Mehta for their mentorship on this project, the XCP-CPW team for hosting this workshop, especially through the difficulties of the time, and Los Alamos National Laboratory and the DOE for funding this work.

Calvin J. Young *is a first year PhD student at Texas A & M University studying Mechanical Engineering. He holds a Bachelor's degree from the University of Missouri in the same field. His research interests are in shock tube experiments, multiphase mixing, and combustion, and is currently in the process of constructing a new facility for the study of multiphase detonations in hydrocarbon fuel droplets and vapors.*

Bibliography

Brown-Dymkoski, Kasimov, and Vasilyev. A characteristic based volume penalization method for general evolution problems applied to compressible viscous flows. *Journal of Computational Physics*, 2013.

Schneider and Vasilyev. Wavelet methods in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 2010.

Wagner. A multiphase shock tube for shock wave interactions with dense particle fields. *Experiments in Fluids*, 2012.

Chapter 4

Deep Neural Networks for Photon and Neutron Transport

Team Members
Le Nyguyen and Jeffrey Keithley

Mentors
Derek Armstrong (XCP-8), Eric Nelson (XCP-8), and Garry Maskaly (XCP-8)

Abstract

Los Alamos National Laboratory (LANL) simulates low altitude ($<$ 10 - 15 km) and ground based EMP (Electromagnetic Pulse) models (E1 and E2 phases) with MCNP and FDTD codes. These models are extremely computationally expensive, taking on the order of weeks to run. The majority of the run-time is in MCNP calculating the energy deposition rate and photocurrent density to feed into FDTD which calculates the EMP's electric field. In this project, we used a multi-layer recurrent neural network to fit existing MCNP energy deposition rate data and then for prediction. Once trained, the recurrent neural network successfully reproduced MCNP results within 1-2% error in a fraction of the time.

4.1 Introduction

MCNP is a Monte Carlo based particle transport simulation code [MCNP-Team (2003)]. A modified version of MCNP provides reliable results for simulating the energy deposition rate and photocurrent density from low altitude ($<$ 10-15 km) photon and neutron sources in air. The photocurrent density and energy deposition rate are then fed into FDTD, a Maxwell solver, that computes the evolution of an EMP's electric field with time. This is very computationally expensive, and a MCNP and FDTD run generally takes on the order of a few weeks to a month to complete, with most of the time taken by MCNP. The purpose of this project is to develop a machine learning based method to model the energy deposition rate and photocurrent density due to photon and neutron sources in significantly less time. An EMP is driven by the photon and neutron weapon output where the gamma rays, being prompt, scattered, or induced by neutrons, ionize the atmosphere via Compton scattering. This model would be used as a surrogate to MCNP and feed the same results into FDTD, making the entire EMP calculation much faster [Yee (1966)]. Because of the time constraints of the project, we were only able to develop a model for the energy deposition rate.

4.2 Background

4.2.1 Electromagnetic Pulse

MCNP and FDTD simulate the EMP effects of a nuclear device detonated in air at low altitudes ($<$ 10-15 km) and on the ground. The detonation in air versus detonation on the ground determines the geometry of the EMP propagation, but fundamentally it is generated in the same way for all nuclear based EMPs. This paper will briefly go over the course grained model developed by Karzas and Latter (WJ Karzas & Richard Latter, 1962; WJ Karzas & R Latter, 1962) and Longmire (Chavin et al., 1979; H. Longley & Longmire, 1972; H. J. Longley & Longmire, 1973; C. Longmire, 1978; Longmire, 1974; C. L. Longmire, 1978; Longmire, 1986, 1987, 1995, 2004; Longmire & Gilbert, 1980; Longmire et al., 1987; Longmire & Longley, 1973), that is widely accepted by the relevant scientific community [Rivera and et al. (2016)].

The behavior of an EMP is broken up into three phases: E1, E2, and E3. E1 spans the first microsecond, E2 spans from one microsecond out to a second, and E3 spans from one second to 10s of seconds. MCNP and FDTD are only used to model E1 and E2 of an EMP and do not have the computational capability to model E3. E1 is created from prompt gamma rays propagating directly from the blast. The gamma rays Compton scatter electrons off atoms in the air that, in mass, continue outward creating this phase of the EMP. E2 is very similar to E1. It is created by scattered and neutron induced gamma rays Compton scattering electrons off atoms in the air. The scattered gammas are from interactions that have already occurred in the atmosphere, while the neutron induced gamma rays are produced by neutrons from the initial blast interacting with nuclei in the air. These intermediate steps in the gamma ray production are what makes the E2 EMP lag behind E1.

There are a few key factors that determine the effect of an EMP, such as the geometry of the electric field. If the electric field is perfectly symmetric, there would be no net current to produce an EMP outside of the source region. Asymmetry is key to producing an EMP and in a low altitude air burst EMP the asymmetry is created by variations in atmospheric density as well as asymmetry because of the air-ground interface. The height of burst of the EMP is another determining factor in its effect. This is partly due to geometric spreading, which is when the intensity of the EMP drops off rapidly as it travels away from the source.

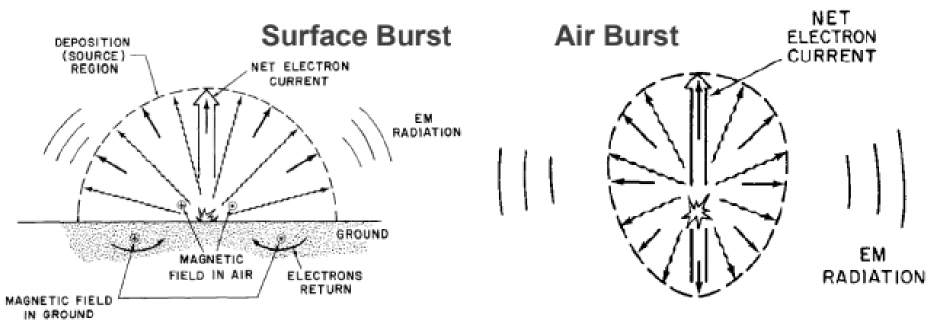


Figure 4.1: Low Altitude EMP, Glasstone and Dolan (1977)

4.2.2 Monte Carlo N-Particle Code (MCNP)

The Monte Carlo N-Particle code developed at LANL is utilized to simulate particle transport behavior relevant for a nuclear burst EMP, calculating energy deposition rate and photocurrent density. The nature of the method used by MCNP is to aggregate many instances of particle behavior instead of deterministically solving the appropriate transport equations [MCNP-Team (2003)]. This results in a long run time due to the computationally intensive process of simulating many particle instances. Despite this, the Monte Carlo method used works well with problems that are difficult to solve deterministically. This method generates the energy deposition rate $(MeV/cm^3/sh)$, which is the energy added to a cell due to particle interactions. MCNP gets the energy deposition per interaction with a look-up table. The energy deposition rate is primarily related to free electrons interacting with air, slowing down, and heating up the air. The energy deposition is fed into FDTD, which is a Maxwell solver that computes the ionization rate, solves for air chemistry, and computes the time-varying electric field. FDTD takes the energy deposition rate and computes an ionization rate by simply assuming an ion pair is created for each 34 eV of deposited energy. MCNP has the ability to model various types of particle interactions, including Compton scattering, the photoelectric effect, and pair production for photons. For neutrons, it has the capability to model interactions such as elastic and inelastic scattering, as well as neutron capture [Reily and et al. (1991)].

Variance reduction in MCNP is used to increase the precision of results from the simulation and avoid consuming additional run time to achieve a smaller error. The prominent methods of variance reduction used by the MCNP calculations in this work are truncation, rouletting, and splitting. The method of truncation operates by eliminating certain data points based on a set of criteria. The main criteria is energy and time values that have little impact on the tallies of interest. Rouletting eliminates particles with a certain probability when they cross a particular boundary, which varies depending on the relevance of the region in the simulation. The last method, splitting, happens when the particle crosses into a region of higher importance. Splitting, as the name implies, is when one particle is replaced by serial particles (split) with the same properties, but reduced weights, to get higher statistics in a region.

4.2.3 Recurrent Neural Networks (RNN)

Neural networks have been at the center of this project since its conception, since they are able to learn the behavior of simulated energy deposition rates. Due to the time series nature of these MCNP results, a recurrent neural network (RNN) is a viable solution [Britz (2015)]. The cornerstone of RNN architecture is the history vector h_t, which is how previous data is remembered. The construction of a node is relatively simple, where the input x_t is concatenated with the history vector h_t, then fed through an activation function to scale the values between -1 and 1. A major drawback of these networks is that previous states are forgotten within a few iterations, therefore the context stored farther back in the sequence is forgotten.

4.2.4 Long Short-Term Memory Networks (LSTM)

The most significant difference between an RNN and LSTM architecture is the complexity of the node. While an RNN is barely a jump from a feed forward network, an LSTM incorporates multiple gates in order to systematically determine whether information is relevant [Olah (2015)]. The underlying feature of the LSTM is the cell state C_t, which consists of what information should be forgotten and what information is relevant. The forget gate feeds the concatenation of the history vector and the input vector, $[h_t{-}1, x_t]$, into a sigmoid function and combines it with the previous cell state C_{t-1}. This information is then combined with a scaled vector representing what values should replace the forgotten ones from the previous step. This is now considered the new cell state C_t and the final step is updating the history vector using a combination of the previous history vector, the input vector, and the newly updated cell state. This cell state and history vector are passed into the next node along with the next input vector.

4.3 Methods

4.3.1 Network Architecture

The RNN architecture used was built in Tensorflow using the Keras API. The specifications of how the network was built are given in table 4.1. As we discussed in the background for recurrent neural networks, a drawback to this type of node is that previous states are soon forgotten. The solution to this problem was solved by utilizing the Long Short-Term Memory architecture, which is better suited for longer term memory.

LSTM Layers	4
LSTM Nodes	128
Dense Layers	1
Dense Nodes	1
Activation	$tanh$
Recurrent	$sigmoid$
Batch Size	64
Learning Rate	Exponential Decay

Table 4.1: Network Architecture

4.4 Data Processing

4.4.1 Feature Engineering

In order to get a better fit to our MCNP data, features were added to give the network more data to fit. The features were the following: average energy was used instead of maximum and minimum energy, column density (amount of air between source and tally location), time squared, and the log of the radial distance to the source of the burst. A rigorous exploration of how these features, or what added features in particular improved the fit was not done, but fit improvement was seen after the features were given to the neural network.

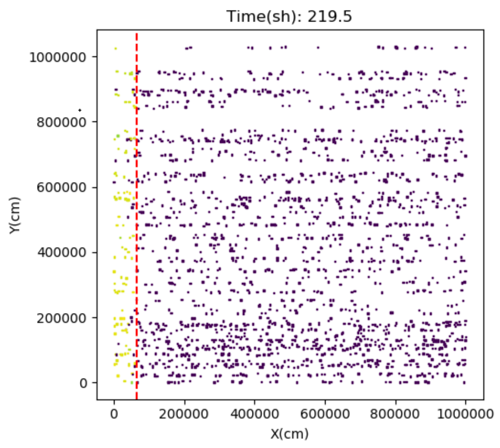
Some of these features are physical in nature, others are simply there to provide more data to fit. The average energy was used in place of the upper and lower bound on the energy range because it is an easier feature to fit than the bounds. The column density is a physical feature; it is a measure of the amount of air from source to tally location. It correlates to the probability of particles making it to a particular location with a given energy, so it is a useful feature for the RNN to use. The log of the radial distance is also another useful physical feature, because the radiative intensity from the source location falls off proportionally to radial distance squared. The log is used in the radial distance due to the energy deposition rate being fit in log-space (i.e., the log of the energy deposition rate is what the RNN is fit against). It will help the RNN predict how many particles should be at a given radius away from the source. The time squared on the other hand is not a physical feature, it does not represent anything in the system, but gives more information the RNN can fit.

Original Features	Original Prediction	Added Features	Final Features	Final Predictions
Height of Burst, Energy Upper Bound, Energy Lower Bound, X Position, Y Position, Time	Energy Deposition Rate	Average Energy, Column Density, t^2, $\log(R)$	Height of Burst, X Position, Y Position, Time, Average Energy, Column Density, t^2, $\log(R)$	$\log$ (Energy Deposition Rate) Zeros Removed

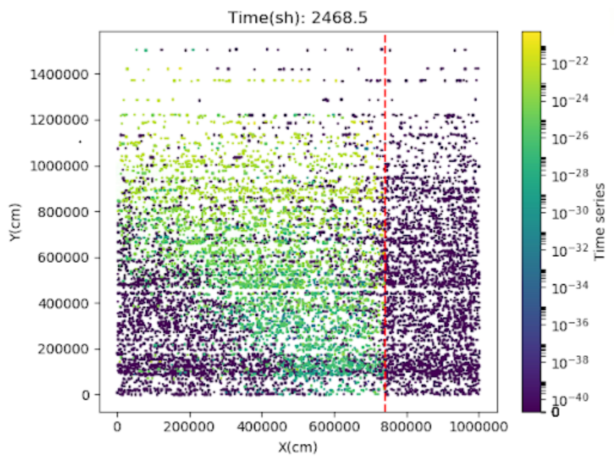
Table 4.2: Original and Processed Features

4.4.2 Data Smoothing

As well as adding features, the log of the energy deposition rate (the prediction) was taken as well. The zeroes in the prediction were also masked out. The energy deposition rate spanned 20 orders of magnitude and the network struggled to fit this, with around 70% of the values being zero, due to the time it took for the particles to populate the tally grid. For fitting to neutron data, the energy deposition rate had to be smoothed in order to fit. The average of every 12 points on each side was taken to achieve this.



(a) Grid at 219.5 sh



(b) Grid at 2468.5 sh

Figure 4.2: Visualization of Features (x, y, time, energy deposition)

4.5 Results

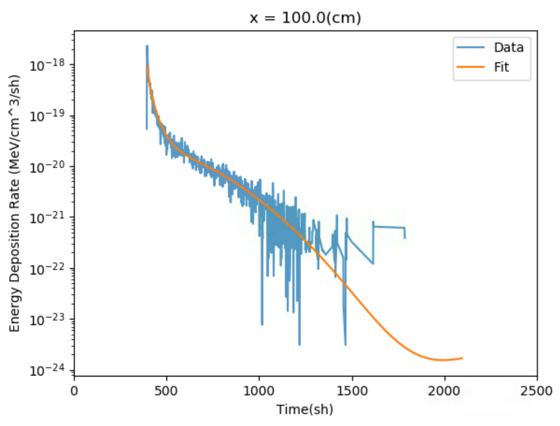
4.5.1 Model Fit

Photon Fit

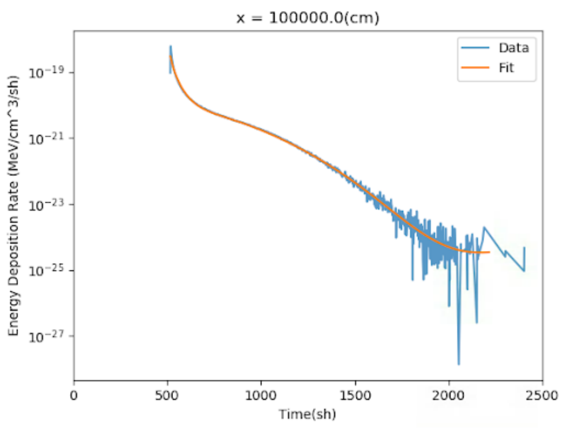
Figure 4.3 shows the RNN fit to the energy deposition rate of our photon data. The photon data was sampled from 50 MCNP runs with a photon energy within the 2-3 MeV range at an unfixed height of burst. The model was set to fit around 2000 shakes at each X location, each with the same Y location.

Height of Burst	X Values	Y value	Energy
$6,135$ m	1 m, 1000 m	$4,950$ m	2.55 MeV

Table 4.3: Photon Model Input Values



(a) 100 Epochs



(b) 100 Epochs

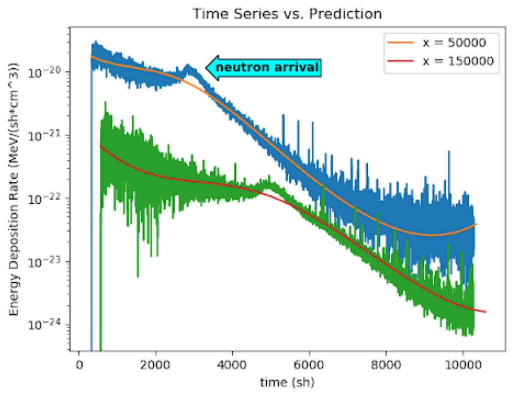
Figure 4.3: Photon Fit Plots

Neutron Fit

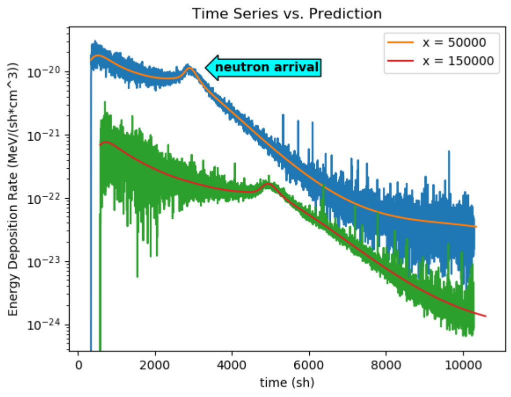
As figure 4.4 clearly shows, the model fits the energy deposition rate for the parameters we specified, except for the neutron arrival times between 3000 and 5000 shakes. This was solved by using smoothed data, a method previously discussed in the data processing section. Even when we train the model in less epochs, the smoothed data still performs drastically better than the original model. The drawback of using smoothed data is that the model takes almost 12 hours to train, as opposed to the model using unsmoothed data, which takes less than 2 hours to train.

Height of Burst	X Values	Y value	Energy
$1,000$ m	500 m, 1500 m	100 m	7.05 MeV

Table 4.4: Neutron Model Input Values



(a) 100 Epochs

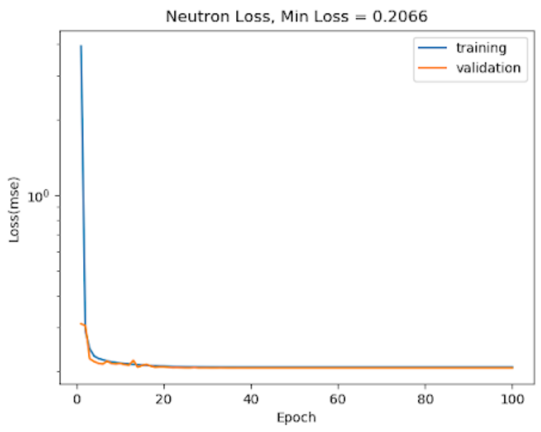


(b) 60 Epochs (Smooth)

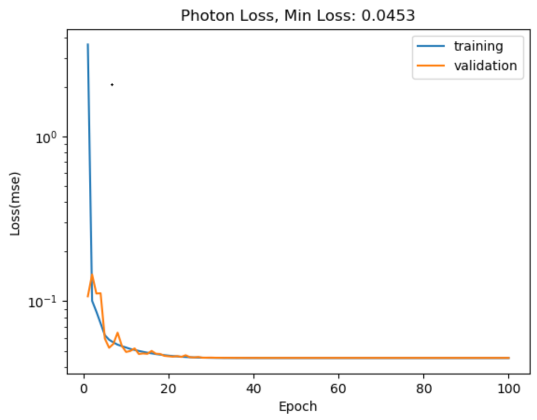
Figure 4.4: Neutron Fit Plots

4.5.2 Training Loss

Figure 4.5 illustrates the loss per epoch during the training time for our models, which show both photon and neutron predictions quickly achieve a low loss (MSE) and converge to a good fit. The loss for the photon produces more noise initially, but settles to a better fit overall by an order of magnitude. The neutron model converges faster, but maintains a higher loss than the photon model. This is due to the differences in complexity of the photon and neutron data. With the neutron data being harder to fit, we would expect a higher loss.



(a) Neutron Training Loss



(b) Photon Training Loss

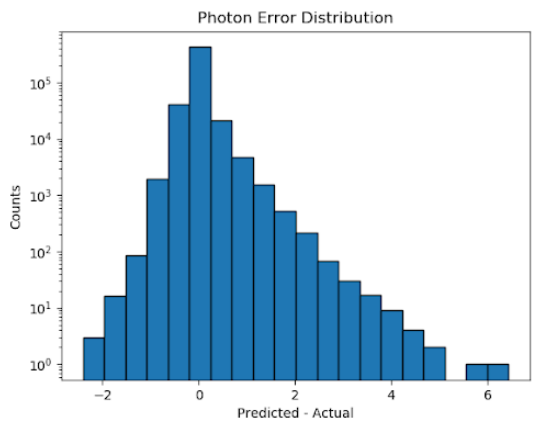
Figure 4.5: Training Loss Plots

4.5.3 Error Analysis

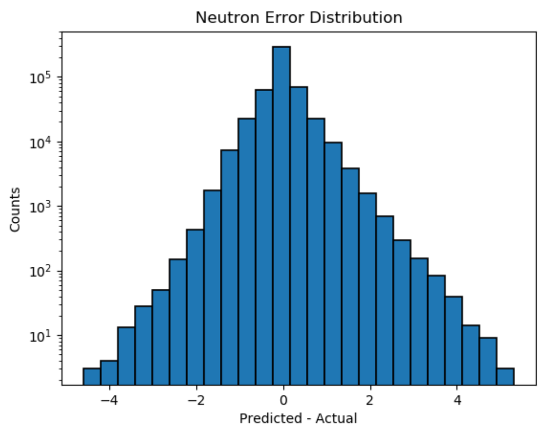
Our neural network fits well to the data with the error being around 1-2%. This error is misleading though, as it is fit to noisy data. Much of the error in the fit is the model ignoring the statistical noise in the MCNP data and does not reflect the accuracy of the fit. As we can see from figure 4.6, the distribution for neutron prediction error $\hat{y} - y$ is symmetric and roughly normal. While we have some outliers, some of these may be attributed to the noise in the actual simulation due to its stochastic nature. From figure 4.6 it may be concluded that the behavior of the outliers is more extreme in the photon prediction model, but values are more centered about 0, so it performs better than the predictor for neutrons in terms of overall fit.

	Photon Fit	Neutron Fit	Neutron (Smooth Data) Fit
MSE (log Space)	0.0452	0.2131	0.1351
RMSE (log Space)	0.2128	0.4616	0.3675
RMSRE	0.00727	0.01811	0.01462

Table 4.5: Fitting Errors



(a) Errors for Photon Fit



(b) Errors for Neutron Fit

Figure 4.6: Error Distribution

4.6 Discussion

It is important to state the neural network we developed is not a replacement for MCNP since it fits to data but cannot generate original EMP results. Our neural network is an extrapolation on current MCNP data, meaning that it can be used to estimate the energy deposition rate from a set of parameters MCNP has not been run on yet based on the sets of parameters data MCNP already has results for. Suppose we want the energy deposition rate over all time and space for a height of burst MCNP had not been initialized at before: We can have our neural network estimate its value in a matter of days instead of performing an entire month long MCNP run.

A back of the envelope calculation was done to estimate how the run time of our RNN compared to the run time of MCNP. The calculation was simple; we took the run-time for our RNN to model a single position and energy at a certain height of burst over all time and multiplied it over all positions and energies. The photon RNN could make a calculation in .08 seconds. There are 2,000 X and Y tallies in a full MCNP grid and 100 energies. This calculation results in the RNN taking about 9,000 CPU hours to run vs. MCNP which takes about 150,000 CPU hours. From this rough calculation we can see that our RNN is about 16.6x faster than MCNP.

4.7 Future Work

Many aspects of this study could be researched more thoroughly, particularly the generalizability of the model. The analysis presented in this paper was specific to a limited energy range for both photon and neutron data with a fixed height of burst. Training models on the entire energy spectrum would be both informative and beneficial for a possible future implementation, as well as smoothing data to fit the neutron arrival time better. The fit of the RNN could have been further optimized with a more thorough hyper parameter search. Parameters such as layers, nodes, learning rate, and batch size were optimized by hand because of time constraints. A grid search or controlled random search should be done to find the parameters that allow the best hyper parameters. Finally, the development of a user-friendly interface that could utilize the trained models without requiring extensive knowledge of the intricacies involved with the method would make the code more usable.

4.8 Conclusion

MCNP along with FDTD simulates low altitude E1 and E2 EMPs. The calculations needed for this take weeks to run with the majority of the run-time being MCNP calculating the energy deposition rate and photocurrent density. To cut down on run-time, a multi-layer recurrent neural network was developed as a way to predict MCNP results of energy deposition rate to feed into FDTD. This was done successfully with RNNs being able to fit photon data and smoothed neutron data. Run time was greatly reduced with the RNN being able to calculate the energy deposition rate for all time, space, and energies in 9,000 CPU hours compared to a recent MCNP run which took about 150,000 CPU hours. The model also fit the data well, with an error of 1-2% mostly due to statistical noise in the MCNP results. The fitting error was mostly Gaussian suggesting that the models were unbiased, but there was a slight skew in the photon error showing it tended to overestimate values. Multi-layer RNNs have shown very promising results as a surrogate to MCNP for FDTD. There is still work to be done as the RNNs are still limited in scope and can be further optimized. Hopefully this project has laid the groundwork for the further exploration of neural networks and other machine learning techniques as solutions to EMP simulation problems.

4.9 Acknowledgments

We would like to thank LANL and the XCP Summer Workshop coordinators for giving us the opportunity to do this project, especially in these unprecedented times (COVID-19 Pandemic). They have gone the extra mile of making this experience totally remote with online collaboration and lectures. We would also like to thank our mentors, Dr. Derek Armstrong, Dr. Eric Nelson, and Dr. Garry Maskaly for the work they did. They guided us through this project and put in the extra work to give us the tools we needed to succeed in spite of not being at the lab.

Le Nguyen *is a senior at Michigan State University majoring in Physics and minoring in Data Science. He is currently looking at graduate programs in Data Science and is interested in computational modeling and algorithm development.*

Jeffrey Keithley *received his bachelor's degree in mathematics from New Mexico Tech and hopes to perform research at LANL for 1-2 years before pursuing a PhD. He is interested in using computational methods and machine learning to improve simulation and solve problems in physics and computer science.*

Glossary

- **Batch:** The set of data passed through the network
- **Column Density:** The amount of air from source to tally location
- **Energy Deposition Rate:** The rate energy is added to a cell due to particle interactions

- **Epoch:** When the entire data set is passed through a network once
- **Iteration:** When one batch is passed through the network
- **Mean Squared Error (RMSE):** $= \frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2$
- **Root Mean Squared Error (RMSE):** $= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\hat{Y}_i - Y_i\right)^2}$
- **Root Mean Squared Relative Error (RMSRE):** $= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\frac{\hat{Y}_i - Y_i}{Y_i}\right)^2}$
- **Variance Reduction:** A method of increasing the precision of a Monte Carlo method without using more time to run

Bibliography

D. Britz. Recurrent neural networks tutorial. *WILDML*, 2015.

S. Glasstone and P. Dolan. The effects of nuclear weapons. *United States Department of Defense*, 1977.

X-5 MCNP-Team. MCNP — A general monte carlo n-particle transport code. *Los Alamos National Laboratory*, 2003.

C. Olah. Understanding LSTM networks. *Colah's Blog*, 2015.

D. Reily and et al. Passive nondestructive assay of nuclear materials. *Los Alamos National Laboratory*, 1991.

M.K. Rivera and et al. EMP/GMD phase 0 report, A review of emp hazard environments and impacts. *Los Alamos National Laboratory*, 2016.

K.S. Yee. Numerical solution of initial boundary value problems involving maxwell's equations in isotropic media. *IEEE*, 1966.

Chapter 5

Emulating Fission Observables

Team Members
Samuel Ozier and Shane Blade

Mentors
Amy Lovell, Ionel Stetcu, and Michael Grosskopf

Abstract

Computational models can help us understand physical phenomena, interpret experimental data, and make predictions where measurements are not possible. Using the Los Alamos National Laboratory (LANL)-developed Cascading Gamma-Ray and Multiplicity for Fission (CGMF) code, neutron multiplicity and average neutron energy were calculated with this simulator as a computational assistant to the experimental findings of fission. CGMF is able to reproduce many properties of this nuclear phenomenon; unfortunately, some of these calculations do not reproduce the experimental data to the desired level of fidelity. Of the observables studied, the average neutron energy predictions from CGMF showed a noticeable divergence from the given experimental data. As a result, a Gaussian Process Regression emulator was implemented to overcome the differences. The values from this emulator greatly improved the quality of CGMF calculation overall with some values still showing discrepancy, especially where the emulator was required to model sharp, local features of the average neutron energy. Future endeavors could involve the expansion of discrepancy modeling for other observables of interest.

5.1 Introduction

5.1.1 Fission

Nuclear fission is a nuclear reaction in which a compound nucleus splits into two or more smaller fragments, or fission products. Furthermore, the production of these lighter fragments is often accompanied by the immediate emission of neutrons and photons (γ rays) during this fission process, prompt neutrons and photons. The release of subatomic particles stems from the fragments' deformation and intrinsic excitation due to the resulting instability produced by the fission. To produce a more stable isomer or ground-state, the fragments will emit prompt neutrons and photons. Once the initial fission products have undergone these prompt emissions, further decays will likely take place. Low-energy fission, e.g. induced by thermal neutrons, can only be caused in the isotopes of plutonium and uranium, whose nuclei contain an odd number of neutrons: ^{233}U, ^{239}Pu, and ^{235}U, for example.

The nuclear fission is also an alternative to carbon based energy sources. To produce energy, the binding energy of the resulting elements must be greater than that of the starting element–when the atoms split, heat is generated , and, subsequently, this heat generates steam. This steam is then used by wind turbines to make electricity. Relevant to the history of Los Alamos National Laboratory, the understanding of nuclear fission, and nuclear physics in general, is critical to the non-proliferation efforts made by the United States government, and other nations around the world.

5.1.2 CGMF

CGMF is a physics-based Los Alamos National Laboratory produced Monte Carlo Hauser-Feshbach code that describes properties of prompt neutrons and γ rays [Becker et al. (2013); Talou et al. (in progress)]. CGMF was developed due to the challenge of executing certain experiments and was designed to calculate the correlations between different fission observables; the goal of CGMF is to simulate fission observables for reactions which are not measured. CGMF can reproduce many prompt properties of nuclei, however one of the challenges is with the neutron energies, specifically. Presently, there are internal physics models that have been studied to address this issue, but another viable option is to use an emulator. The purpose of the emulator would be for discrepancy modeling, i.e. we use the emulator as a tool to overcome the discrepancy between CGMF calculations and data obtained from Chi-Nu [Kelly (2020)] (average outgoing neutron energy) and ENDF/B-VIII.0 [Brown et al. (2018)] (average number of emitted neutrons).

5.2 Methods

5.2.1 CGMF: Fission Simulator

Based on the Fission Fragment Decay [Talou (2010)] and Cascading Gamma-ray and Multiplicity [Kawano and Holloway (2011)] codes from prior LANL scripts, CGMF was ”developed to model the de-excitation of the fission fragments on an event-by-event basis, following the successive emissions of neutrons and photons”. It uses the Hauser-Feshbach statistical theory of compound nuclear reactions [Hauser and Feshbach (1952)] to produce the probabilities of particle emission at each step of the cascade. As for Monte Carlo, it simulates the decay of excited fission fragments via prompt neutron and γ-ray emission and then records and analyzes its histories. It is a Monte Carlo because the initial conditions of the fission fragments are sampled for each event, and the nucleus can decay in a different manner each time (e.g. two neutrons and three gammas, or one neutron and four gammas, etc). The development of this code has allowed for heightened levels of predictions and observations into correlations between the fission fragments, neutrons, and photons, as in [Talou et al. (2018)].

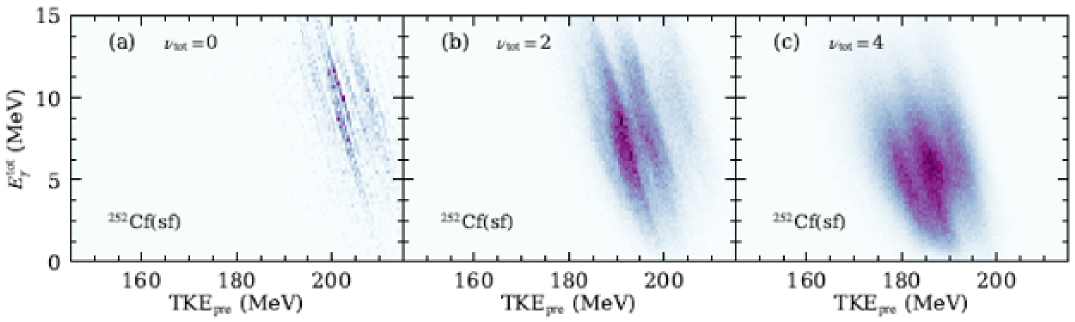
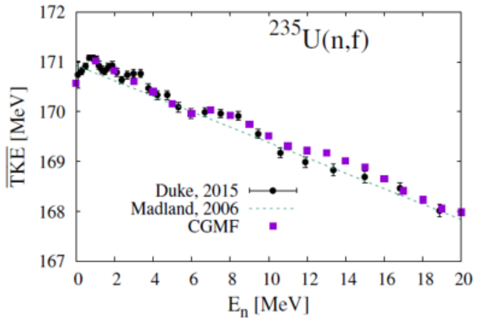
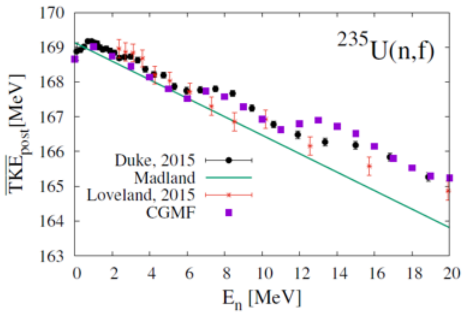


Figure 5.1: Total γ-ray Energy vs. Total Fragment Kinetic Energy. Figure from [Lovell et al. (2019)].



(a) Pre-Neutron Emission



(b) Post-Neutron Emission

Figure 5.2: Total Kinetic Energy of Fission Fragments (MeV) vs. Incident Energy (MeV). Figure from [Stetcu et al. (2017)].

The average prompt fission neutron multiplicity, the prompt fission neutron multiplicity distribution, the average prompt fission neutron multiplicity as a function of mass, charge, and kinetic energy, among other properties, can be modeled using CGMF calculations. For the purposes of our project, we were focused on the average prompt neutron multiplicity and average outgoing neutron energy.

5.2.2 Emulator: Gaussian Process Regression

Gaussian process (GP) models are non-parametric probabilistic models that find a distribution over functions consistent with observed data [Rasmussen and Williams (2006)]. As a regression model, a GP acts similarly to linear regression or neural networks. GPs are more flexible in capturing complex, nonlinear responses than linear regression. Although neural networks are capable of reproducing these nonlinear responses, they struggle to provide a well-founded prediction uncertainty, whereas GPs can provide this. Another advantage of Gaussian Processes is that the kernel functions can be specified, in addition to being able to customize said kernel functions.

Kernel functions, also called covariance functions, are important features of Gaussian Processes, and thus our emulator, as they largely determine the smoothness and shape of the prior and posterior of the Gaussian Process. Kernel functions control the similarity between each pair of training points. They also assume the training points have the same target values. Kernel functions can be distinguished by two classes: stationary and non-stationary. Stationary covariance functions are only concerned with the distance between the input values, whereas non-stationary kernel functions also are considerate of the values of the specific data-points themselves. One example of a kernel function, the Matérn stationary function, is specified as follows:

$$\kappa(x_i, x_j) = \frac{1}{\Gamma(v)2^{v-1}} \left(\frac{\sqrt{2v}}{l} d(x_i, x_j) \right)^v K_v \left(\frac{\sqrt{2v}}{l} d(x_i, x_j) \right) \tag{5.1}$$

where v controls the level of smoothness, $d(.,.)$ is the euclidean distance between points, $K_v(.)$ is a modified Bessel function, and $\Gamma(.)$ is the gamma function. Note that when v, the term that ultimately controls smoothness, approaches an arbitrarily large value, it results in the squared exponential covariance function. The Matérn function can thus take several forms. In particular, it can take different forms when $v = \frac{1}{2}$, $v = \frac{3}{2}$, and $v = \frac{5}{2}$. The different Matérn functions (and the Squared Exponential) are given below:

$$\text{Squared Exponential: } \kappa(x_i, x_j) = \sigma^2 \exp\left(-\frac{(x_i - x_j)^2}{2l^2} \right) \qquad v = \infty \tag{5.2}$$

$$\text{Matérn 1/2: } \kappa(x_i, x_j) = \exp\left(-\frac{1}{l} d(x_i, x_j) \right) \qquad v = \frac{1}{2} \tag{5.3}$$

$$\text{Matérn 3/2: } \kappa(x_i, x_j) = \left(1 + \frac{\sqrt{3}}{l} d(x_i, x_j) \right) \exp\left(-\frac{\sqrt{3}}{l} d(x_i, x_j) \right) \qquad v = \frac{3}{2} \tag{5.4}$$

$$\text{Matérn 5/2: } \kappa(x_i, x_j) = \left(1 + \frac{\sqrt{5}}{l} d(x_i, x_j) + \frac{5}{3l} d(x_i, x_j)^2 \right) \exp\left(-\frac{\sqrt{5}}{l} d(x_i, x_j) \right) \qquad v = \frac{5}{2} \tag{5.5}$$

For further clarification, κ is the kernel function that measures the similarity between any training input x_i and unlabeled input x_j. The l term is the length scale parameter; where it can either be a scalar value or a vector, and can have the dimension as the inputs x. The σ^2 term in Squared Exponential determines the average distance of your function away from its mean.

5.3 Results

5.3.1 CGMF: Initial Findings

In any Monte Carlo simulation, it is important to make sure that the results have converged - that is, a large enough number of events has been calculated. Using the most direct approach to this problem, the highest number of events possible would provide the most adequate predictions; however, more events require more time for the calculation to run. Therefore, a balance between number of events and time of calculation is needed. Initial running of CGMF indicated a roughly linear relation between the time taken to complete calculations and the number of events requested. Too low of an event count would likely result in unconverged results, but simply increasing the count would cause the process of acquiring statistics to take too long when a high number of points for the different incident energies would need to be calculated. Consequentially, an analysis was performed to see this diminishing variance as the number of events was increased. Figure 5.3 illustrates how the average neutron energy across an incident energy range for certain numbers of events differ from the same property when 6.4 million events were calculated.

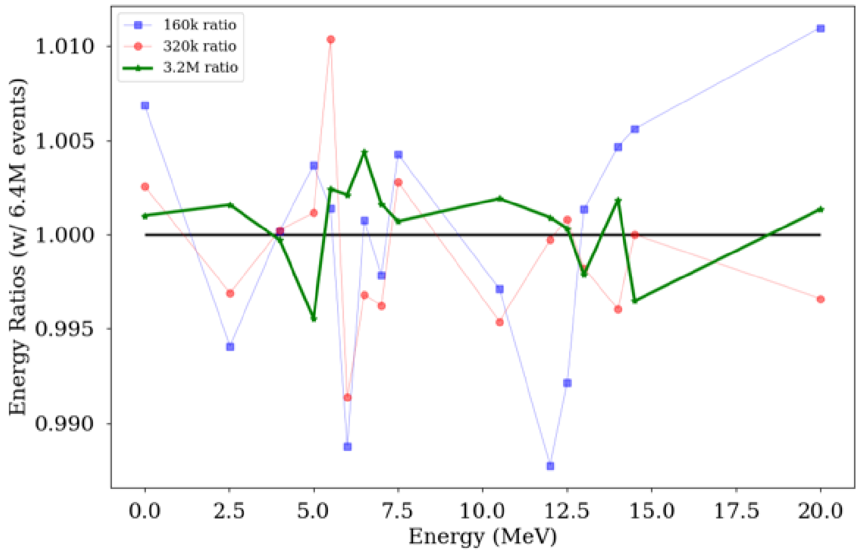
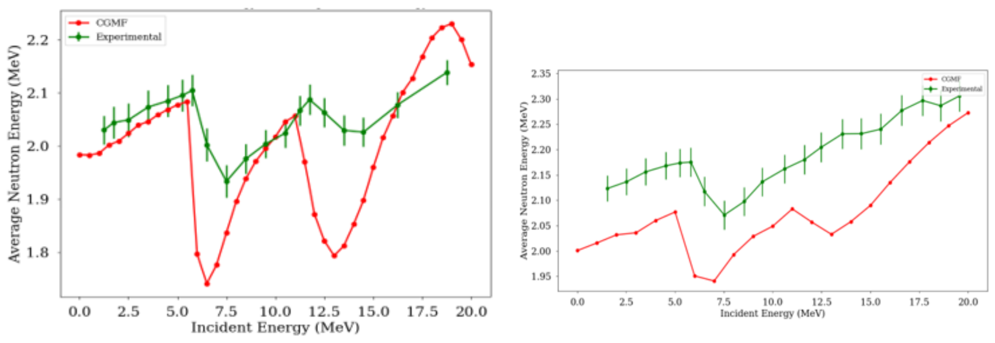


Figure 5.3: Average Neutron Energy when 160k (blue), 320k (red) and 3.2M (green) events were simulated as a ratio to 6.4 M events as a function of Incident Neutron Energy (MeV)

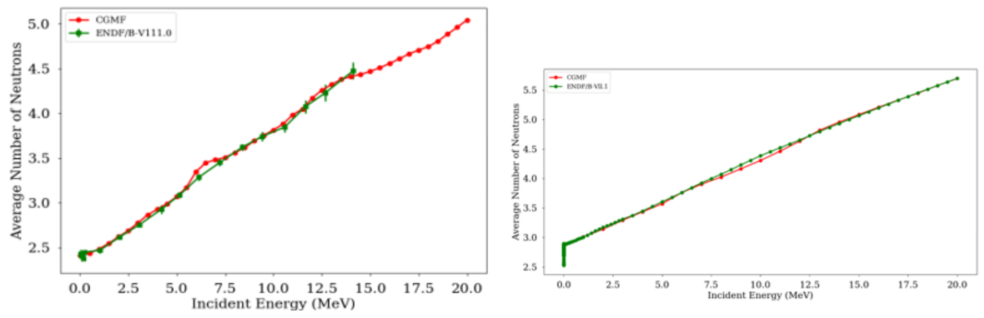
Looking at how the ratios of the average neutron energy of an increasing number of events compared to 6.4 million events, an overall convergence was present, as expected. Even with 160 thousand events, most of the plotted points are within 1% of the 6.4 million event baseline. A simulated run with 160 thousand events would only take from 10 to 15 minutes, so higher event counts were calculated to see if a noticeably higher convergence could be achieved while not taking too long. Eventually, 3.2 million events was chosen; as Figure 5.3 shows that this number has a great fidelity to 6.4 million events while not taking hours to run a simulation. As a result, all of the CGMF calculations were run with 3.2 million events for the rest of the project.

As the project's focus was both neutron multiplicity and average neutron energy over an incident neutron energy range from 0 to 20 MeV, CGMF simulations were performed for both ^{235}U and ^{239}Pu. The predictions were then compared to experimental data. For the neutron multiplicity, the ENDF/B-VIII.0 evaluation [Brown et al. (2018)] was the experimental comparison. Preliminary Chi-Nu experimental data [Kelly (2020)] served as the comparison for average neutron energy.



(a) ^{235}U (b) ^{239}Pu

Figure 5.4: Average Neutron Energy (MeV) vs. Incident Energy (MeV). Green curve shows the experimental data compared to the CGMF calculations in red.



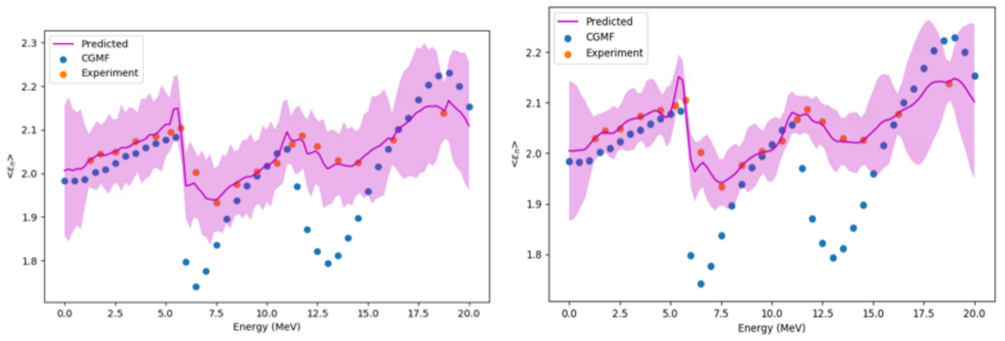
(a) ^{235}U (b) ^{239}Pu

Figure 5.5: Average Number of Neutrons vs. Incident Energy (MeV). Green curve shows the ENDF/B-VIII.0 evaluation compared to the CGMF calculations in red.

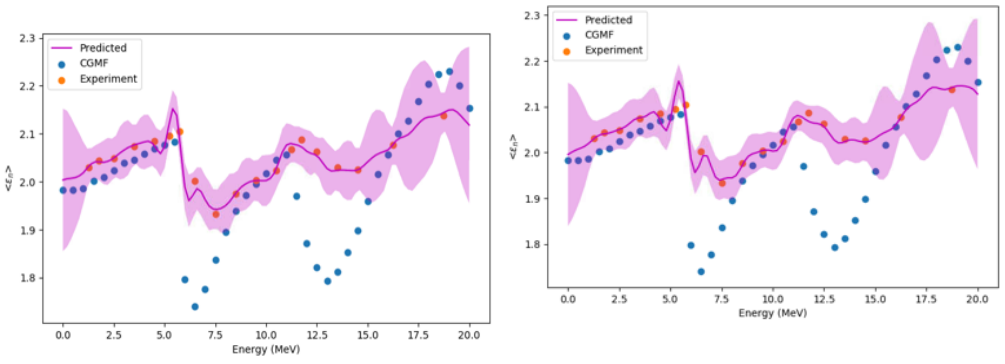
As illustrated in Figs. 5.4 and 5.5, the degree to which the CGMF simulator is able to reproduce the experimental information varies significantly. In Figure 5.5, the ENDF/B-VIII.0 evaluation can be closely compared to the CGMF calculation for both nuclei. In contrast, in Figure 5.4, the CGMF calculation compared to the Chi-Nu experiment shows that a noticeable discrepancy is present across the incident energy range for both of the analyzed nuclei; however, the behavior of the predicted models with respect to each nuclei and their differences is also noteworthy. Figure 5.4a has the CGMF calculation closely resembling what Chi-Nu measured, staying near the region of error provided, until 5 MeV. Above that incident energy, the experimental average neutron energy has a drop-off and shows a resulting physical phenomenon where a neutron is likely emitted before fission occurs. The consequence of this event would take away energy from the fission event, and similar drop-offs later in the incident energy spectrum could be explained by this phenomenon. In a response to this sharp decrease in average neutron energy, the simulator also decreases its value at the same incident energies; however, the predicted drop-offs are much more severe compared to the Chi-Nu measurement. For Figure 5.4b, the calculations from CGMF are consistently lower than the experimental counterpart and fall outside of the experimental error bar. The overall shape of the ^{239}Pu calculations, however, shows a higher fidelity than the ^{235}U predictions. As a result of these illustrations, additional methods of shoring up the discrepancy for the simulator's average neutron energy predictions are needed. Fortunately, emulators exist as one approach for discrepancy modeling. As an effort to utilize an emulator, a Gaussian Process Regression model was chosen to better the predictive modeling for average neutron energy.

5.3.2 Introducing the Emulator

When using a Gaussian Process Regression model to help mitigate the discrepancy between CGMF and experimental data, one important factor that influences the GP's performance is its covariance function, which can change the modeling capability of the emulator in reproducing the training set and making predictions. For the project, Matérn and Squared Exponential covariance functions were studied to test the performance differences between these functions. A visual comparison is made in Fig. 5.6.



(a) Matérn 12 (b) Matérn 32



(c) Matérn 52 (d) Squared Exponential

Figure 5.6: Kernel functions ^{235}U. The orange points are the training data, the blue points are CGMF, and the purple line is the prediction when all training data is included.

Looking at the results of using the emulator for ^{235}U, the GP model shows in improved reproduction of the Chi-Nu data compared to CGMF across the board. For each kernel function, the prediction follows both the shape and the magnitude of average neutron energy closely with a standard deviation that displays the uncertainty. With these covariance functions, however, the behaviors of the mean prediction and the general shapes of the uncertainty present do vary. Figure 5.6a, unlike the other figures present, shows a more jagged uncertainty interval (purple shaded bands) compared to the other kernel function and its mean value (solid purple line) matches that behavior. Such a behavior could be a symptom of over-fitting. To work on a confirmation of whether this concern is valid, a quantification for the emulator's fitting could be implemented. An evaluation of a model's goodness of fit was chosen to be the chi-squared value, (χ^2), defined as

$$\chi^2 = \sum_{i \in g} \left(\frac{y_i - \hat{y}_i}{\sqrt{\sigma_{E,i}^2 + \sigma_{g,i}^2}} \right)^2 . \tag{5.6}$$

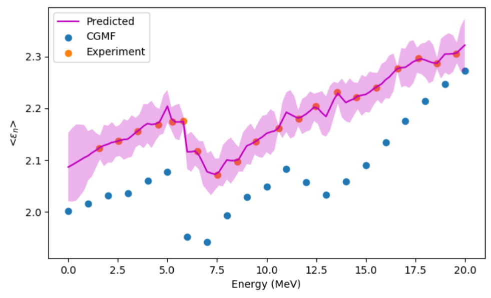
In general, the χ^2 value compares the difference between the experimental data and the emulator's predictions within the overall uncertainty present, here from both the experiment and the emulator. Due to the nature of GP regression, targeting a specific value, e.g. 1, to represent an ideal fit is a bit nebulous due to the degrees of freedom at play. The primary purpose of this calculated value was to compare the kernel functions studied and the nuclei analyzed for the project. The sets i and g are the sets of average neutron energy values across the incident neutron energy spectrum for the experiment and emulator respectively. y_i is an average energy value for a given incident energy from the Chi-Nu experiment, while $\hat{y}_i$ is the corresponding emulator value. Finally, $\sigma_{E,i}$ is the experimental uncertainty for the i^{th} experimental value, and $\sigma_{g,i}$ is the emulator's uncertainty for the i^{th} predicted average neutron energy.

This calculation included all of the points from the experimental data to perform a direct comparison to the predicted model. No points were taken out at this point from the emulator's input, as an initial analysis was performed without cross validation. These χ^2 values are listed in Table 5.1 for each of the listed kernel functions.

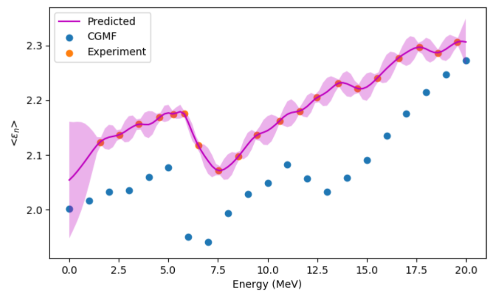
Kernel Function	χ^2
Matérn 1/2	0.891
Matérn 3/2	1.566
Matérn 5/2	1.826
Squared Exponential	2.238

Table 5.1: χ^2 values without cross validation for ^{235}U with the different kernel functions.

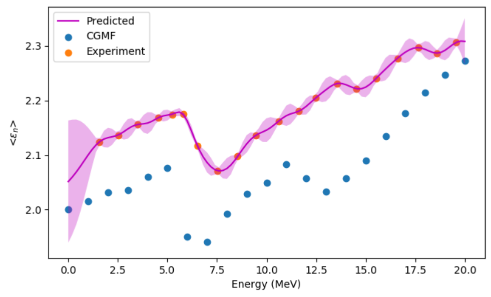
In comparing the calculated χ^2 values, the lowest value comes from the Matérn 1/2 function. Notably lower than the other kernel functions' χ^2s, this function's value provides additional insight into how relative χ^2 values relate how the emulator behaves for each covariance function. The apparently larger standard deviation for Matérn 1/2 likely plays a role for this result, as the χ^2 for the other functions are all closer to 2 than Matérn's value and their prediction models look more similar to each other as in Figure 5.6 Furthermore, the low χ^2 for Mattérn 1/2 could indicate possible over-fitting, as having too low of a χ^2 value can indicate such. Unfortunately, as stated prior, an ideal χ^2 was not yet determined during the project, so this prediction has not been fully verified. Looking at the behavior of the emulated curve, its standard deviation, and the χ^2 value, circumstantial evidence is present for further investigation. Using this method for goodness of fit for the emulator modeling, the same analysis was performed on ^{239}Pu, shown in Fig. 5.7.



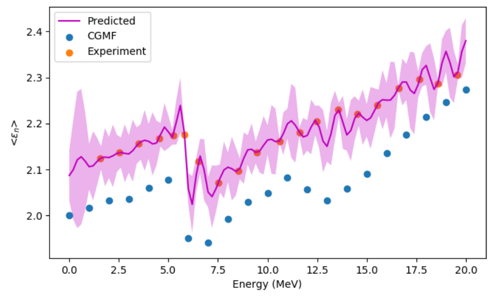
(a) Matérn 1/2



(b) Matérn 3/2



(c) Matérn 5/2



(d) Squared Exponential

Figure 5.7: Kernel functions ^{239}Pu. The orange points are the training data, the blue points are CGMF, and the purple line is the prediction when all training data is included.

The results from Fig. 5.7 show greater variability between each kernel function when compared to ^{235}U; this indicates that different data sets can respond much differently to the choice of kernel function. Of these kernel functions, Matérn 1/2 and Squared Exponential show the greatest amount of over-fitting and selection bias, whereas the prediction for Matérn 3/2 and 5/2 are much smoother. The error bands reflect this as well, where a wider error band indicates more uncertainty for the predictive model. To perform a more quantitative analysis of over-fitting and selection bias, their χ^2 values are displayed in Table 5.2.

Kernel Function	χ^2
Matérn 1/2	102.608
Matérn 3/2	129.557
Matérn 5/2	131.696
Squared Exponential	82.179

Table 5.2: χ^2 values without cross validation for ^{239}Pu with the different kernel functions.

One can see immediately that the χ^2 values for ^{239}Pu are much larger than the χ^2 values for ^{235}U (Table 5.2). This is because, in addition to the χ^2 value being a measure of fit, there is also the consideration of uncertainty. The tighter uncertainty intervals for ^{239}Pu are seen both at the incident energies where there is experimental data and in between the experimental data points; overall, there is a higher confidence in the prediction. The narrower uncertainty intervals between the experimental data points for ^{239}Pu compared to ^{235}U show there is a higher confidence for these predictions

5.3.3 Implementing Cross Validation

To test the robustness of our emulator, we implemented the statistical method of cross validation. This process is done by breaking the data into four random groups, and then leaving one of those groups out of the training each time. Figure 5.8 shows the iterative process of sub-setting our data and running the emulator code for Matérn 3/2 for ^{239}Pu.

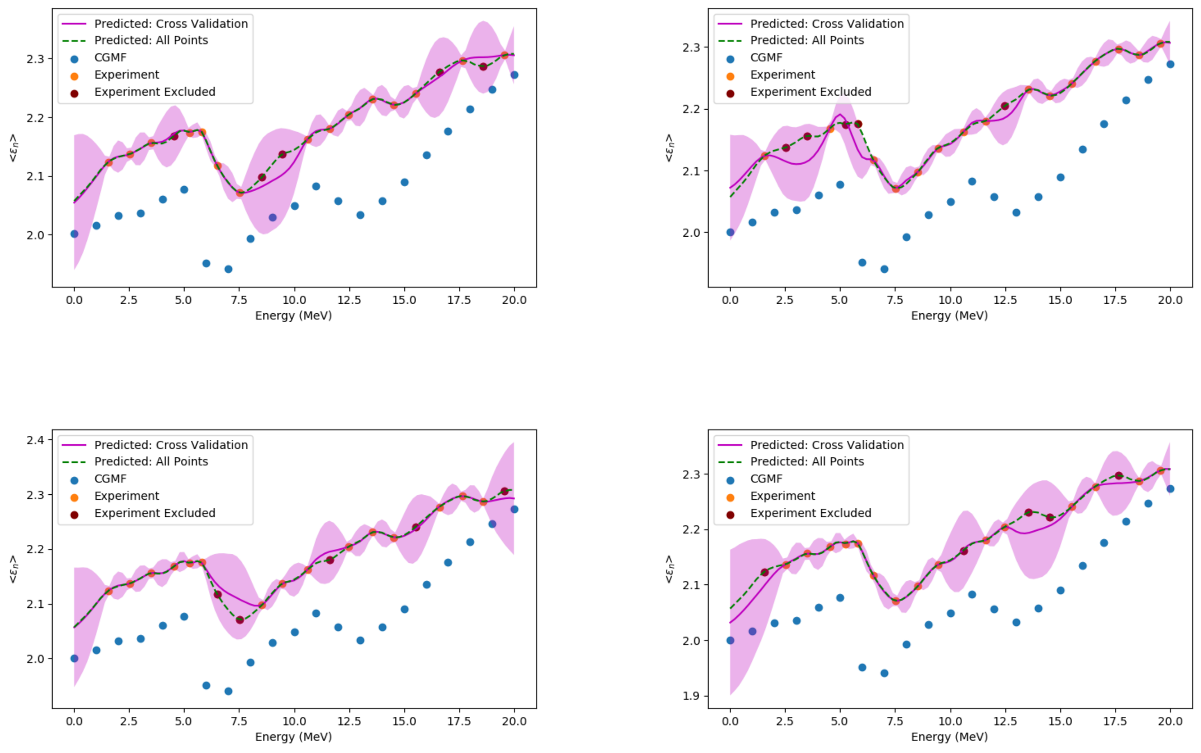


Figure 5.8: Cross validation with Matérn 3/2 for ^{239}Pu. The maroon points are the testing data, orange points are the training data, the green dashed line is the prediction for when all points are included in the training, and the purple line is the prediction when the points in orange are removed from the training set.

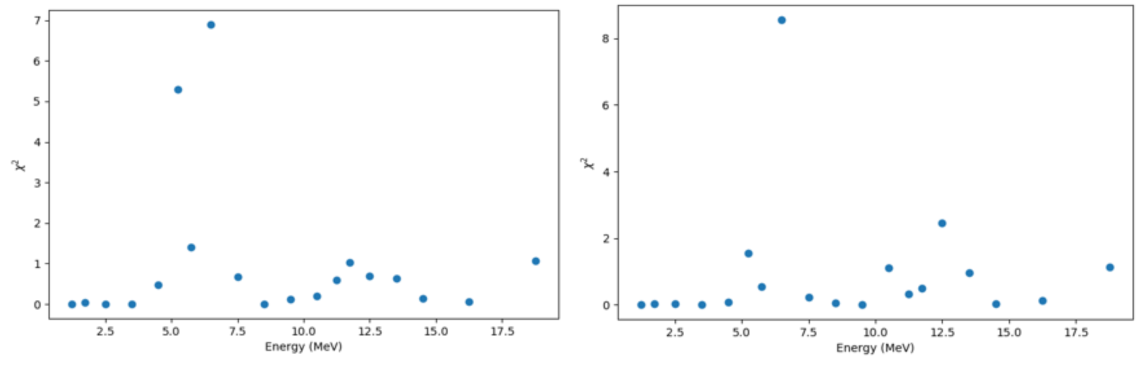
As the plots show, when data is left out, the uncertainty bands grow much larger, which is what one would expect. Additionally, the GP does have a tendency to over-fit as well; it gravitates towards the CGMF calculation as opposed to reproducing the experimental Chi-Nu data. In Table 5.3, we give the χ^2 values of the energy points left out of the training set.

Test Data	χ^2
[8, 17, 2, 11, 16]	38.215
[12, 1, 15, 4, 10]	24.298
[3, 18, 19, 13]	33.413
[6, 9, 7, 14, 5]	20.405

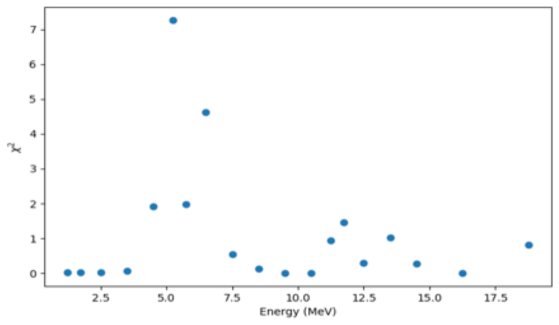
Table 5.3: The left column represents the test data, or the index points excluded from the experimental data, and the right column represents the associated χ^2 value specifically for the test data. The test data in the table is associated with Figure 5.8 counterclockwise starting from the top left plot.

5.3.4 Discrepancy Evaluation

When performing cross validation on ^{239}Pu, the discrepancy between the GP prediction and the experimental values varies based on which incident energy points were removed. As a result, χ^2 values vary greatly across the incident energy range. The next step to see where these larger differences originated, a χ^2 was evaluated for each individual Chi-Nu incident energy. Using this method would allow for a more in-depth analysis as to which incident energies provide the greatest difficulty for the emulator to predict when not included in the training set. For the sake of time, only ^{235}U was studied in this manner.



(a) Matérn 5/2 (b) Matérn 3/2



(c) Squared Exponential

Figure 5.9: χ^2 with cross validation for all experimental points for ^{235}U. Each χ^2 was evaluated by removing the corresponding Chi-Nu energy point and comparing the produced model to the missing Chi-Nu energy point.

Figure 5.9 shows the relationship between the incident energy and the resulting discrepancy when that single energy point from the experimental set is removed from the emulator training set. Most notably, heightened values of χ^2 around 5 MeV exist for all three tested kernel functions. A local maximum also occurs around 12 MeV, those the behaviors for each of the covariance functions is shown to be less consistent than the 5 MeV maximum. With Figure 5.9, the incident energy values near 5 MeV are shown to provide the greatest discrepancy when left out of the prediction model and, furthermore, provide the greatest difficulty for the emulator to predict.

Looking at Figure 5.4 and Figure 5.6, one can take note of the step drop-off, as previously noted in comparing the CGMF predictions to Chi-Nu experimental data. This sharp, local feature is much more severe in its alterations to average neutron energy compared to other segments of incident energy, resulting in the larger drop of fitting quality for the emulator compared to the rest of the incident energy values.

5.4 Conclusion

For this project, we studied neutron-induced fission of ^{239}Pu and ^{235}U and analyzed how well CGMF predicts properties of neutron-induced fission. Important in multiple scientific topics such as astrophysics and non-proliferation, comprehension of induced fission holds importance for current research. Producing an effective predictive model would work to both cover for the limitations of experimentation and help understand computational physics models themselves. CGMF serves as an initial step into creating such a model.

For CGMF, we observed a linear relation between the number of fission events observed and the time taken to compute them. Of the number of fission events, 3.2M events showed minimal variation to an increased number of events per task. While neutron multiplicity well reproduced the evaluated values from ENDF/B-VIII.0, the average neutron energy showed greater discrepancy compared to experimental measurements from Chi-Nu.

Regarding the Gaussian Process, we tested different kernel functions on both ^{239}Pu and ^{235}U: Matérn 3/2, Matérn 5/2, Matérn 1/2, and Squared Exponential. For ^{239}Pu, the kernel functions that performed best were Matérn 3/2 and Matérn 5/2, whereas Matérn 1/2 and Squared Exponential showed signs of selection bias and over-fitting. For ^{235}U, the kernel functions that worked best were Matérn 5/2 and Squared Exponential, while Matérn 3/2 and Matérn 1/2 were not as good. Also, between the two isotopes, ^{239}Pu showed more variance in overall fitting when shifting between kernel functions, whereas ^{235}U was more consistent; this demonstrates how different data sets can respond to different kernel functions, even if the same observables are being studied.

Based on which indexed incident energies were excluded from the experimental set, the quality of Gaussian Process prediction varied greatly. The emulator struggled to capture sharp, local features; sudden drops in average neutron energy near an incident neutron energy of 5 MeV had the highest divergence from the observed points in cross validation.

For future work, the discrepancy model could be used to model other observables of interest and other fission reactions. Additionally, it would be interesting to see the emulator being used in more than one dimension, i.e. it would be able to calculate the average outgoing neutron energies and the average number of neutrons simultaneously.

5.5 Acknowledgements

The students of this project would like to thank Amy Lovell, Ionel Stetcu, and Michael John Grosskopf for their time and incredible patience as they mentored during the summer, as well as Daniel Israel, Madison Andrews, Garry Maskaly, and Katrin Hammerling for their efforts in making this experience a reality.

Sam Ozier *is a graduate student at Mississippi State University. He graduated with a Bachelor's in Chemical Engineering at the same university and is currently pursuing a Master's in Computer Science. At the university, Sam is employed by its Center for Advanced Vehicular Systems (CAVS) to develop on autonomy for off-road vehicles. Born in the state of Mississippi, Sam enjoys playing guitar, reading, and sports in addition to long walks on the beach.*

Shane Blade *is a rising junior at Clemson University studying physics (BS). His research interests include nuclear physics, AMO physics, and non-linear dynamics; he enjoys the interdisciplinary aspects of these subfields, e.g. the importance of nuclear physics in the field of astrophysics. In terms of hobbies, he enjoys listening, playing and creating music.*

Bibliography

B. Becker, P. Talou, T. Kawano, Y. Danon, and I. Stetcu. Monte carlo hauser-feshbach predictions of prompt fission γ rays: Application to $n_{\text{th}} + {}^{235}$u, $n_{\text{th}} + {}^{239}$pu, and ^{252}cf (sf). *Phys. Rev. C*, 87:014617, Jan 2013. doi: 10.1103/PhysRevC.87.014617. URL `https://link.aps.org/doi/10.1103/PhysRevC.87.014617`.

D.A. Brown, M.B. Chadwick, R. Capote, A.C. Kahler, A. Trkov, M.W. Herman, A.A. Sonzogni, Y. Danon, A.D. Carlson, M. Dunn, D.L. Smith, G.M. Hale, G. Arbanas, R. Arcilla, C.R. Bates, B. Beck, B. Becker, F. Brown, R.J. Casperson, J. Conlin, D.E. Cullen, M.-A. Descalle, R. Firestone, T. Gaines, K.H. Guber, A.I. Hawari, J. Holmes, T.D. Johnson, T. Kawano, B.C. Kiedrowski, A.J. Koning, S. Kopecky, L. Leal, J.P. Lestone, C. Lubitz, J.I. Márquez Damián, C.M. Mattoon, E.A. McCutchan, S. Mughabghab, P. Navratil, D. Neudecker, G.P.A. Nobre, G. Noguere, M. Paris, M.T. Pigni, A.J. Plompen, B. Pritychenko, V.G. Pronyaev, D. Roubtsov, D. Rochman, P. Romano, P. Schillebeeckx, S. Simakov, M. Sin, I. Sirakov, B. Sleaford, V. Sobes, E.S. Soukhovitskii, I. Stetcu, P. Talou, I. Thompson, S. van der Marck, L. Welser-Sherrill, D. Wiarda, M. White, J.L. Wormald, R.Q. Wright, M. Zerkle, G. Žerovnik, and Y. Zhu. Endf/b-viii.0: The 8th major release of the nuclear reaction data library with cielo-project cross sections, new standards and thermal scattering data. *Nuclear Data Sheets*, 148:1 – 142, 2018. ISSN 0090-3752. doi: https://doi.org/10.1016/j.nds.2018.02.001. URL `http://www.sciencedirect.com/science/article/pii/S0090375218300206`. Special Issue on Nuclear Reaction Data.

Walter Hauser and Herman Feshbach. The inelastic scattering of neutrons. *Phys. Rev.*, 87:366–373, Jul 1952. doi: 10.1103/PhysRev.87.366. URL `https://link.aps.org/doi/10.1103/PhysRev.87.366`.

Toshihiko Kawano and Shannon Holloway. Cgm ver.3. Technical report, Los Alamos Nat. Lab. LA-CC-11-018, 2011.

K. J. Kelly. private communication. *private communication*, 2020.

A. E. Lovell, I. Stetcu, P. Talou, G. Rusev, and M. Jandel. Prompt neutron multiplicity distributions inferred from γ-ray and fission fragment energy measurements. *Phys. Rev. C*, 100:054610, Nov 2019. doi: 10.1103/PhysRevC.100.054610. URL `https://link.aps.org/doi/10.1103/PhysRevC.100.054610`.

C. E. Rasmussen and K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

I. Stetcu, P. Talou, and T. Kawano. Prompt fission neutron and properties as a function of incident neutron energy. *EPJ Web Conf.*, 146:04026, 2017. doi: 10.1051/epjconf/201714604026. URL `https://doi.org/10.1051/epjconf/201714604026`.

P. Talou, R. Vogt, J. Randrup, M. E. Rising, S. A. Pozzi, J. Verbeke, M. T. Andrews, S. D. Clarke, P. Jaffke, M. Jandel, T. Kawano, M. J. Marcath, K. Meierbachtol, L. Nakae, G. Rusev, A. Sood, I. Stetcu, and C. Walker. Correlated prompt fission data in transport simulations. *The European Physical Journal A*, 54(1):9, Jan 2018. ISSN 1434-601X. doi: 10.1140/epja/i2018-12455-0. URL `https://doi.org/10.1140/epja/i2018-12455-0`.

P. Talou, T. Kawano, I. Stetcu, P. Jaffke, M. E. Rising, and A. E. Lovell. Fission fragment decay simulations with the cgmf code. *Computer Physics Communication*, in progress.

Patrick M. Talou. Fd, ”fission fragment decay”, version 1.0. Technical report, Los Alamos Nat. Lab. LA-CC-10-003, 2010.

Chapter 6

Materials Phase Diagrams from Density Functional Theory

Team Members
Quinn White and Elise Koskelo

Mentors
Daniel Sheppard, Daniel Rehn, Ann Mattson

Abstract

As materials with a diverse set of structural, electrical, and thermal properties, 2D group IV monochalcogenides have gained significant interest in the condensed matter community for their applications in electronics and thermoelectrics. In this work, we study the structural phase transition of monolayer tin sulfide (SnS) using density functional theory. Specifically, we investigate three different methods for driving the transition between the rectangular monolayer phase (a single exfoliated layer of the bulk *Pnma* crystal) and the square *Cmcm* phase (a single exfoliated layer of the bulk *Cmcm* crystal) including electrostatic gating, mechanical strain, and temperature. In the electrostatic gating setup, a SnS monolayer is exposed to excess charge in a parallel plate capacitor. Here, we find that a transition voltage of -8.78 V can be used to induce the structural phase transformation. This method was also applied to monolayer GeSe, finding a transition voltage of -6.75 V. For the SnS monolayer, we find that compression along the b-axis above six percent of the *Pnma* phase is enough to drive the transition. In addition, we studied the lattice dynamics of the SnS monolayer using the finite displacement method, valid at zero temperature, and molecular dynamics simulations at finite temperature using the Temperature Dependent Effective Potential (TDEP) method, to account for the strong anharmonicities of the *Cmcm* phase. By computing the phonon modes of the monolayer, we confirm the existence of a displacive phase transition of the *Cmcm* to the *Pnma* phase upon cooling and estimate a monolayer transition temperature in the range of 240 K. Given the large array of methods presented here to drive the structural phase transition, SnS presents a promising material for phase change memory and thermoelectric devices. Combining these methods could result in even more efficient methods for controlling the phase transition.

6.1 Introduction

6.1.1 Motivation

In this work, our aim was to study structural phase transitions in monolayer SnS between its low energy *Pnma* phase, and high energy *Cmcm* phase, which is equivalent to the thermal average of the two instances of the rectangular *Pnma* phase. We studied driving this transition via three separate techniques, namely electrostatic gating, mechanical strain, and temperature. Work investigating the application of these methods to other 2D materials has been performed in recent years, suggesting the potential for these methods to be used to drive phase transformations in monolayer SnS (Rehn et al., 2018a; Duerloo et al., 2014; Wang et al., 2017; Zakhidov et al., 2020; Rehn et al., 2018b).

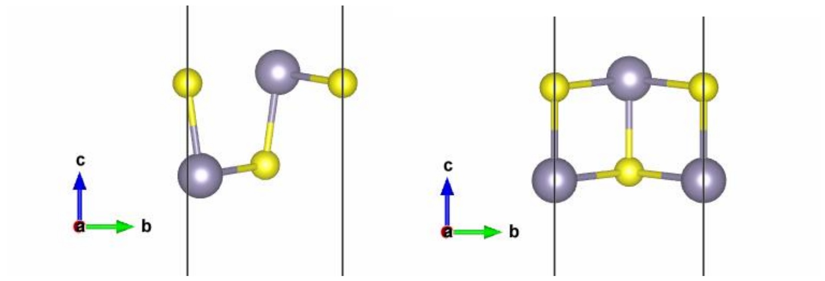


Figure 6.1: Unit cells of the two structural phases of 2D SnS. Left: Low temperature, rectangular, *Pnma* phase. Right: High temperature, square, *Cmcm* phase. The high temperature phase is structurally equivalent to the thermal average of the two instances of the low temperature phase. Note that the vertical lines represent periodic boundary conditions.

The ability to drive the structural phase transition in SnS efficiently has important applications in electronics and thermoelectrics. For example, SnS could be used as a material for phase change memory, which could require a low energy cost and fast transition rate between the two phases (Rehn et al., 2018a). Alternatively, the lower thermal conductivity of the high energy *Cmcm* phase could be used as the basis for a high power thermoelectric material (Skelton et al., 2016).

6.1.2 Density Functional Theory (DFT)

In order to investigate the phase transitions of monolayer SnS, we use density functional theory (DFT). Before discussing what DFT is, it is good to first explain why we need such a theory. In quantum mechanics, we know that all of the system properties are encoded within the systems wave function Ψ. Once one has the wave function, it is possible to calculate the probability amplitude associated with observables such as energy or momentum, which you can use to make predictions for experiments. However, In order to determine Ψ, it is necessary to solve the Schrödinger equation. If we are only concerned with the electronic structure of our system, the general many-body

Schrödinger equation can be written as [1] (Capelle, 2006)

$$\left[\sum_{i}^{N}\left(-\frac{\hbar^2\nabla_i^2}{2m}+v(\mathbf{r}_i)\right)+\sum_{i<j}U(\mathbf{r}_i,\mathbf{r}_j)\right]\Psi(\mathbf{r}_1,\mathbf{r}_2,...,\mathbf{r}_N)=E\Psi(\mathbf{r}_1,\mathbf{r}_2,...,\mathbf{r}_N) \tag{6.1}$$

where $v(\mathbf{r})$ describes how the N electrons interact with any nuclei, while $U(\mathbf{r}_i, \mathbf{r}_j)$ is needed for electron-electron interactions. Unfortunately, this problem is nearly impossible to solve for anything beyond hydrogen-like systems, due to large number of degrees of freedom introduced by the electron-electron interactions.

Hohenberg-Kohn Theorem

Given that we are unable to solve the many-body Schrödinger equation for anything but the most simple cases, we need another method for performing calculations. One such approach is density functional theory, which proposes that instead of trying to solve for the ground state wave functions which will then be used to calculate probability amplitudes for observables, begin with a specific observable and use it to find the wave functions. This specific observable is the electron density $n(\mathbf{r})$, which can be written as:

$$n(\mathbf{r})=N\int d^3\mathbf{r_2}\int d^3\mathbf{r_3}...\int d^3\mathbf{r_N}\Psi^*(\mathbf{r},\mathbf{r}_2,...,\mathbf{r}_N)\Psi(\mathbf{r},\mathbf{r}_2,...,\mathbf{r}_N) \tag{6.2}$$

The reasoning behind this idea is provided by the **Hohenberg-Kohn Theorem**, which states that the electron density $n(\boldsymbol{r})$ contains all of the information needed to determine the ground state properties of a system (Ann Mattson, 2020). Mathematically, this means that our ground state wave function Ψ_0, and thus all associated observables, are unique functionals of the ground state density, $\Psi_0[n_0]$ (Hohenberg and Kohn, 1964). This realization is what allows us to take an incredibly complex problem with many degrees of freedom, and map it down to one depending on only a single vectorial quantity. (Capelle, 2006)

The Kohn-Sham Equations

In order to compute the charge density $n(\mathbf{r})$ in density functional theory, we solve a set of equations known as the Kohn-Sham equations. They are a set of nonlinear eigenvalue equations which must be solved self-consistently (Capelle, 2006):

$$\left(-\frac{\hbar^2}{2m}\nabla^2+V_{eff}(\boldsymbol{r})\right)\psi_\nu=\epsilon_\nu\psi_\nu,\ \nu\in\{1,2,...,N\} \tag{6.3a}$$

$$n(\boldsymbol{r})=\sum_{\nu=1}^{N}|\psi_\nu(\boldsymbol{r})|^2 \tag{6.3b}$$

$$V_{eff}(\boldsymbol{r})=V(\boldsymbol{r})+\int\frac{n(\boldsymbol{r'})}{|\boldsymbol{r}-\boldsymbol{r'}|}d\boldsymbol{r'}+\frac{\delta E_{xc}[n(\boldsymbol{r})]}{\delta n(\boldsymbol{r})} \tag{6.3c}$$

[1]Here we are using the Born-Oppenheimer approximation and assuming our system is non-relativistic.

The simplification of these equations is to treat N interacting electrons in a material as instead N non-interacting "Kohn-Sham" particles with wavefunctions given by $\psi_\nu(\boldsymbol{r})$, and energies ϵ_ν, where $\nu = \{1, 2, ..., N\}$. The energy operator for each state is comprised of the standard kinetic energy operator $-\frac{\hbar^2}{2m}\nabla^2$ and an effective potential V_{eff} as shown in Equation 6.3a). The first two terms of $V_{eff}(\boldsymbol{r})$ in Equation 6.3c) represent the classical Coulombic interations between the atomic nuclei and the electrons, $V(\boldsymbol{r})$, and the electron-electron repulsion, $\int \frac{n(\boldsymbol{r'})}{|\boldsymbol{r}-\boldsymbol{r'}|}d\boldsymbol{r'}$.

All of the energy due to the non-Coulomb interactions between electrons is captured in an **exchange correlation functional**, $E_{xc}[n(\mathbf{r})]$, which is a functional of the charge density. The last term in the effective potential of Equation 6.3c) represents the functional derivative of the exchange correlation function $E_{xc}[n(\mathbf{r})]$ with respect to the charge density $n(\mathbf{r})$. In the case where an analytical expression for the exchange correlation functional were known, the solutions to the Kohn-Sham equations would be equivalent to the that of the many-body Schrodinger equation. In reality, a "divine" correlation functional is not known and computational scientists use empirically-determined functionals which have been developed using assimilation of various data sets. For example some common functionals include Ceperley-Alder (CA) (Ceperley and Alder, 1980) and Ceperley-Alder, with the parametrization of Perdew-Zunger (PZ) (Perdew and Zunger, 1981), which are local density approximations in that they are solely functionals of the charge density $n(\boldsymbol{r})$, and Perdew-Burke-Ernzerhof (PBE) (Perdew et al., 1996), which is a GGA functional in that it is also a functional of the *gradient* of the charge density, $\nabla n(\boldsymbol{r})$ (Ann Mattson, 2020).

The typical procedure for a DFT calculation is to: 1) choose a functional based on the type of calculation and/or material, 2) use an initial estimate for the charge density based on the electronic configuration of the material, 3) calculate a new effective potential based on the charge density and exchange-correlation function, 4) solve the Kohn-Sham equations for the Kohn-Sham orbitals/wavefunctions, 5) update the charge density via Equation 6.3b) and repeat.

All periodic DFT calculations were performed with the Vienna Ab initio Simulation Package (VASP), version 5.4.4 (Kresse and Furthmüller, 1996).

6.2 Phase Transition via Electrostatic Gating

Recent theoretical work has shown that electrostatic gating can be used to drive a structural phase transition in 2D transition metal dichalcogenides (TMDs) such as $MoTe_2$ (Li et al.). Gating-driven transitions were later investigated in experiments, where the transition between the 2H and 1T' phases of $MoTe_2$ was found to occur at transition voltages within the range that was predicted in the theoretical work (Wang et al., 2017; Zakhidov et al., 2020). We now look to apply these techniques to monolayer group IV monochalcogenides, with a particular focus on SnS and GeSe.

6.2.1 Background

In our calculations, we assume a system configuration matching that of figure 6.2. Given this capacitor setup, the total energy of the system may be written as

$$E(Q) = E^I(Q) + \frac{Q^2}{2C} + E^{II}(-Q) \tag{6.4}$$

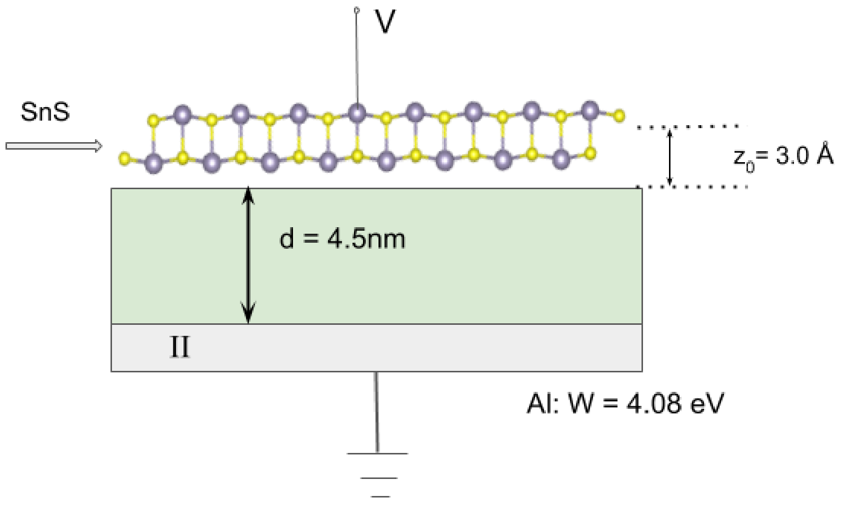


Figure 6.2: Capacitor setup for electrostatic gating. The distance from the center of the SnS monolayer to the surface of the dielectric is $z_0 = 3.0$ Å. The dielectric is chosen to have a thickness of 4.5 nm and sits atop a plate of aluminum, which has a work function of $W = 4.08$ eV.

where $E^I(Q)$ is the energy of our charged monolayer, $Q^2/2C$ is the energy stored in our dielectric medium which has some capacitance C, and $E^{II}(-Q)$, which describes the energy of the metal plate furthest from the monolayer. To calculate the energy of the dielectric, we need only know it's thickness and dielectric constant k. The energy of the plate can also be approximated as

$$E^{II}(-Q) = -QW \tag{6.5}$$

where W is the work function of the metal for the plate. Now all that remains is to calculate the energy of the charged monolayer.

Calculating the Monolayer Energy

The first step in calculating the energy of our charged monolayer is to structurally relax both phases using VASP. Following this, various DFT simulations are run for the monolayer, where different values of charge have been added or removed from the material. In these simulations, a background potential is automatically introduced by VASP to account for the charge that has been added onto the monolayer. Due to the periodic boundary conditions of our system, if this background potential was not included, the added charge would lead to infinite energies arising in our calculation. However, this also means that the energy that is returned is not useful in helping us understand how the charge has modified the monolayers energy. To account for this then, we must subtract off the introduced potential to get accurate results. To do this, we first write the monolayer energy as

$$E^I(Q, z_{ref}, z_f) = E_0 + \int \Delta V(Q', z_{ref}, z_f)dQ' \tag{6.6}$$

where E_0 is the energy of the neutral monolayer, and $\int \Delta V(Q', z_{ref}, z_f)dQ'$ defines the energy needed to move charge from some reference plane z_{ref} to a point z_f, where the total potential is

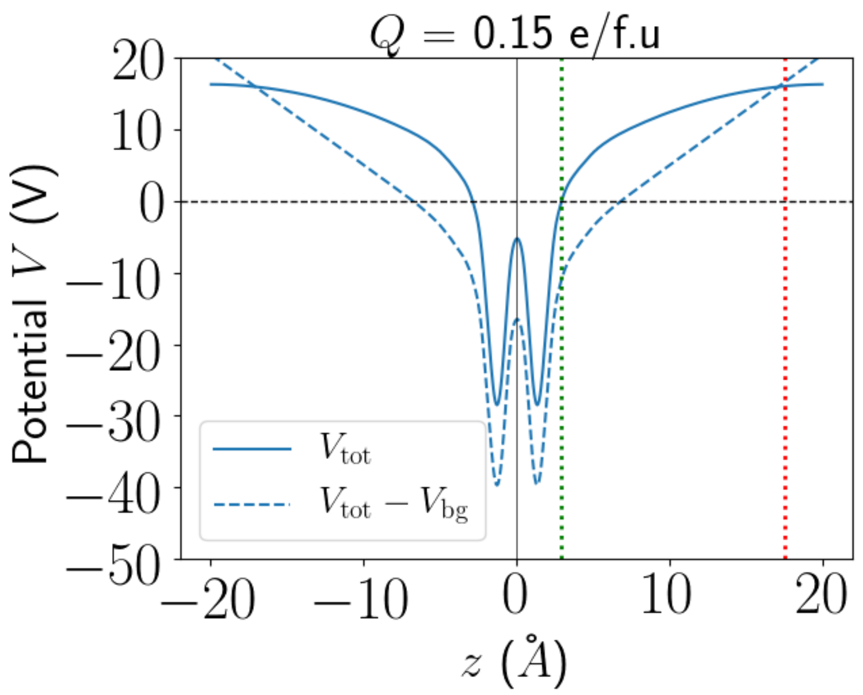


Figure 6.3: Total potential from VASP vs the Potential after the contribution from background charge has been removed. The red line represents our reference plane z_ref, while the green line represents z_f.

equal to the Fermi energy μ_f. It should be noted that while we have chosen to use the Fermi energy in our definition for z_f, this approach will hold for other monolayer band energies, which can be useful in some situations [2]. In it's entirety, we may write $\Delta V(Q', z_{ref}, z_f)$ as:

$$\Delta V(Q', z_{ref}, z_f) = \Big(V_{tot}(Q', z_{ref}) - \mu_f(Q')\Big) - \Big(V_{bg}(Q', z_{ref}) - V_{bg}(Q', z_f(Q')\Big) + \frac{Q'}{2\epsilon_0 A}(z_{ref} - z_f(Q')) \tag{6.7}$$

where V_{bg} represents the background potential introduced by VASP, and A is the area of our monolayer. To calculate the background potential, we apply a result from classical electrostatics. By modeling our monolayer as a charged plate with a uniform charge density, we obtain the following expression for calculating the background potential for a computational cell of length L[3]:

$$V_{bg}(Q', z) = \frac{Q'}{2\epsilon_0 AL}\left(z^2 - \frac{1}{4}L^2\right) \tag{6.8}$$

With the background potential now accounted for, all that is left is to include the terms necessary for computing the magnitude of the uniform electric field that exists within the capacitor resulting from the charged monolayer. This contribution is handled by the last terms in equation 6.7.

Before we can calculate the total energy, we need an analytic expression for $\Delta V(Q', z_{ref}, z_f)$, as this will allow us to complete the integral in 6.6. To determine an analytic expression, we take the

[2]See Band Structure and Electrostatic Gating for a more detailed discussion of this point.

[3]This computational cell length is the same as the value chosen for the c lattice constant.

data obtained from the DFT simulations for our chosen range of Q' to compute the $\Delta V(Q', z_{ref}, z_f)$ values for our choice of reference plane z_{ref}. Once we have an array of $\Delta V(Q', z_{ref}, z_f)$ values corresponding to each Q', a first-degree polynomial is fit to the data using least squares regression. This results in the analytical expression:

$$\Delta V(Q', z_{ref}, z_f) = a_0 + a_1 Q' \tag{6.9}$$

This expression still requires some modification, as it assumes we will be moving charge directly from the reference plane z_{ref} to our monolayer. In reality, we will have another reference plane, which we will call z_0, that is much closer to our monolayer[4] Å. To account for our new reference plane, we treat z_0 and z_{ref} as a parallel plate capacitor, which results in the equation:

$$\Delta V(Q', z_0, z_f) = \Delta V(Q', z_{ref}, z_f) + \frac{Q'}{\epsilon_0 A}(z_0 - z_{ref}) \tag{6.10}$$

which we can then combine with 6.9 to obtain our final analytical expression:

$$\Delta V(Q', z_0, z_f) = a_0 + \left(a_1 + \frac{z_0 - z_{ref}}{\epsilon_0 A}\right) Q' \tag{6.11}$$

Inserting this equation into 6.6 and integrating then gives us our expression for the total energy of our monolayer:

$$E^I(Q, z_0, z_f) = E_0 + a_0 Q + \frac{1}{2}\left(a_1 + \frac{z_0 - z_{ref}}{\epsilon_0 A}\right) Q^2 \tag{6.12}$$

Having computed the monolayer energy, we may now return to equation 6.4 to compute the total energy of our system.

6.2.2 Band Structure and Electrostatic Gating

The goal of this section is to briefly describe how the gating process is able to drive a structural phase transition in these materials. The process can be understood by looking at the electronic states of each phase. As electrical charge is added to the once neutral monolayer, the new charge will look to occupy the lowest energy states available (Wang et al., 2017). However, due to the difference in band gaps between the two phases, as seen in figure 6.4, these states in the conduction band are at a higher energy in the *Pnma* phase compared to the *Cmcm* phase. So, as charge is added, it becomes more energetically favorable for the material to be in the *Cmcm* phase, and the transition occurs.

It is because of this that we are interested in materials that have a large band gap difference, while also having a small difference in energy between phases, as both of these quantities would suggest a transition occurring closer to ambient conditions (Li et al.). Comparing these quantities for the group IV monochalcogenides, we see that SnS and GeSe seem to be the best candidates to undergo a gating driven phase transition, as they have larger differences between band gap energies, while still having a low difference between their phase energies.

[4]Usual values for z_0 range from $3 - 5$ Å, compared to the much larger z_{ref} value of 17.5.

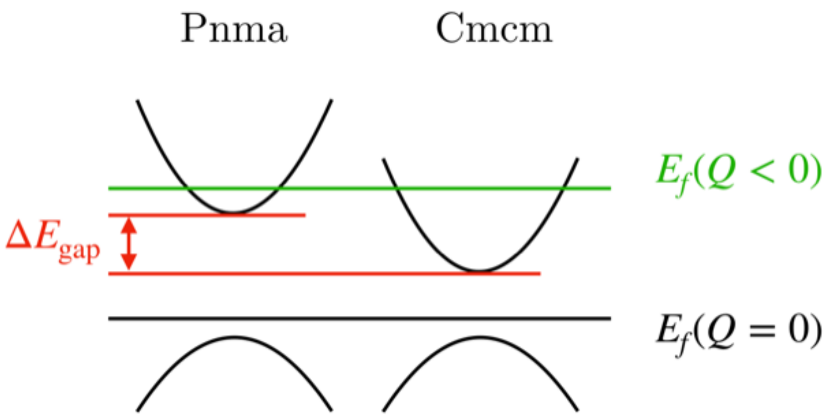


Figure 6.4: Due to the difference in band structure between phases, electrostatic gating is able to be used to drive structural phase transitions.

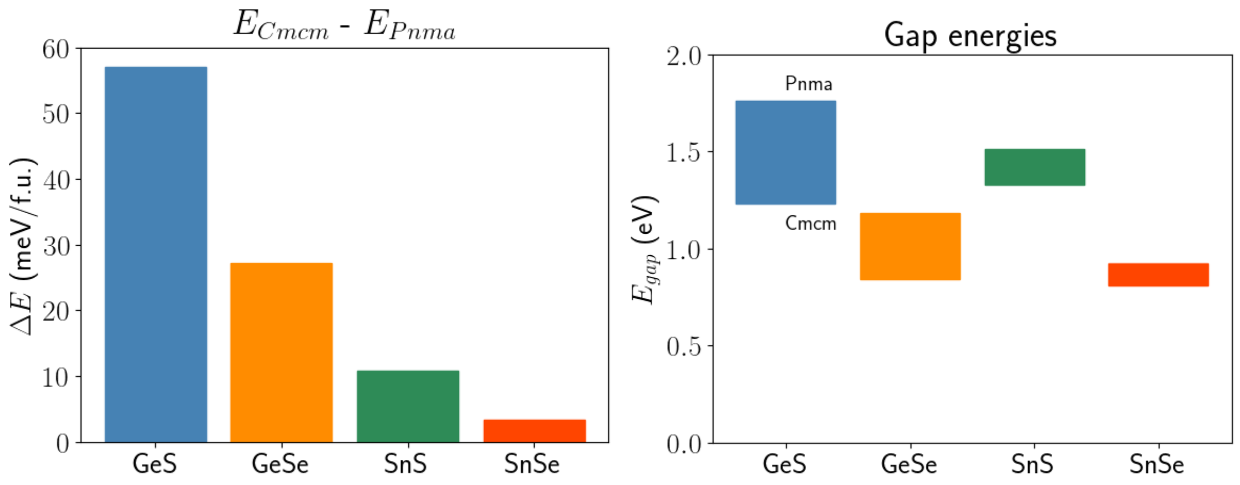


Figure 6.5: Figures showing the difference in energy between phases (left) and the phase band gap energies (right) for each material. All energies and band gaps were computed using the GGA PBE exchange-correlation functional (Perdew et al., 1996).

6.2.3 Gating Driven Phase Transition of SnS

All DFT simulations made use of the projector augmented-wave method with a 600 eV kinetic energy cutoff, and the electron exchange-correlation interaction was treated by the generalized gradient approximation (GGA) functional of Perdew, Burke, and Ernzerhof (Blöchl, 1994; Perdew et al., 1996). An 18x18x1 Monkhorst-Pack k-point mesh was used for sampling the Brillouin zone, and all structural relaxation was done using the conjugate gradient algorithm (Monkhorst and Pack, 1976). Both phases were initially relaxed until an energy convergence of 1×10^{-5} eV could be met within three ionic steps, while a convergence of 1×10^{-8} eV was set for calculations on the charged monolayer. Gaussian smearing of 50 meV was used, and a 40 Åvacuum space was introduced along

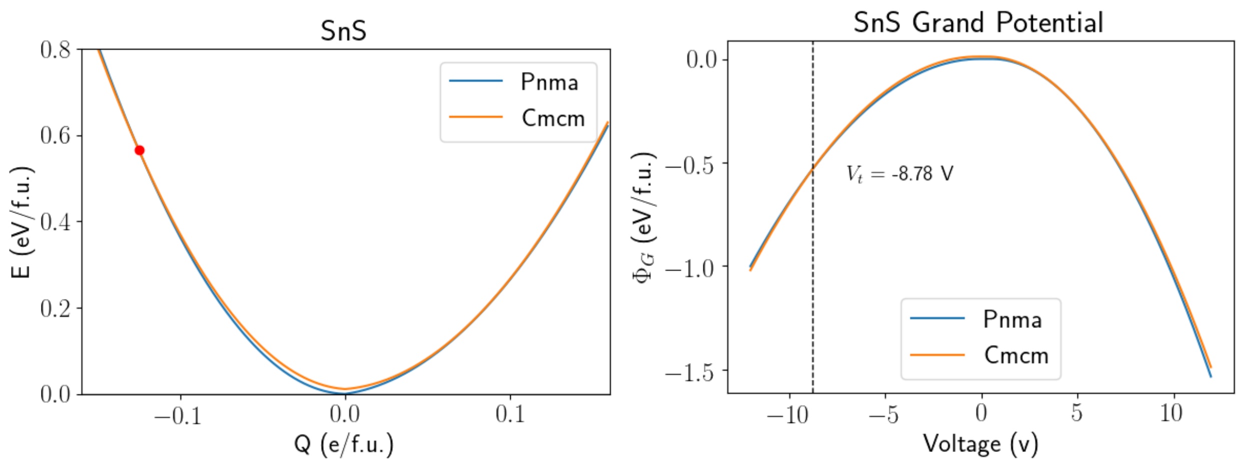


Figure 6.6: Left: Total capacitor energy E as a function of charge Q. We see that under an accumulation of charge, a transition is seen. Right: *Pnma* and *Cmcm* grand potential's Φ_G as a function of gate voltage V. The dashed line marks the intersection of the two curves, which also defines the transition voltage.

the z axis.

For the gating setup, the initial reference plane z_{ref} was chosen to be 17.5 Å, while the secondary plane z_0 was set at 3 Å. Our dielectric thickness was set at 4.5 nm, with a dielectric constant of $k = 25$, and our Q' values varied from -0.03 to 0.15 e/f.u. in increments of 0.01 e/f.u.. Finally, the dielectric was atop an aluminum plate with a work function of $W = 4.08$ eV.

Applying the methods described above, we predict a transition from the *Pnma* to *Cmcm* phase in monolayer SnS will occur at a transition voltage of $V_t = -8.78$ V. The first point of note is that this transition is only predicted to occur under an accumulation of charge, as opposed to existing in both directions. Additionally, the calculated transition voltage is significantly larger than those found in a similar study done for monolayer $MoTe_2$, where transition voltages of -1.8 V and 4.4 V were predicted, and later experimentally confirmed (Li et al.; Wang et al., 2017; Zakhidov et al., 2020).

6.2.4 Gating Driven Phase Transition of GeSe

As a continuation of the calculations done for SnS, simulations were also run for monolayer GeSe. While GeSe is very similar to SnS, it does have a larger difference in bandgap at $\Delta E_{gap} = -0.341$ eV[5]. Given this difference, and the discussion provided in 6.2.2, we would expect to find a lower transition voltage needed for GeSe.

All simulation conditions were identical when compared to the SnS gating calculations, with the only difference being the POSCAR and POTCAR files. As a result, we predict a transition driven by gating in monolayer GeSe to be seen at a gate voltage of $V_t = -6.75$ V. While this is still larger than the transition voltages needed for $MoTe_2$, it is lower than the SnS result.

[5]Compare to SnS value of $\Delta E_{gap} = -0.184$ eV

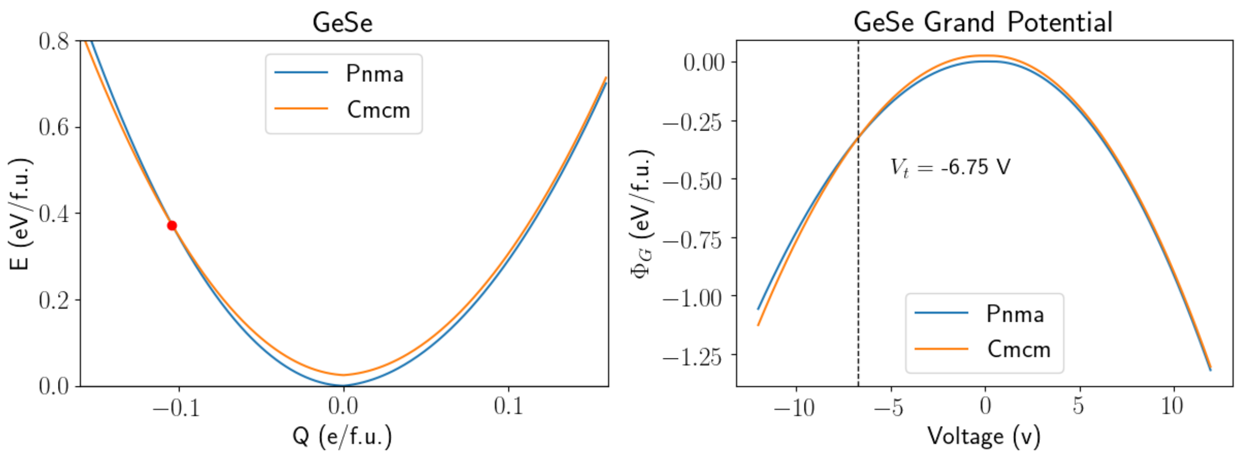


Figure 6.7: Left: Total capacitor energy E as a function of charge Q. Right: *Pnma* and *Cmcm* grand potential's Φ_G as a function of gate voltage V.

Calculating the Grand Potential

To determine all transition voltages, we first need to calculate the grand potential of the system $\Phi_G(Q,V)$ (Li et al.). Here, we define $\Phi_G(Q,V)$ as

$$\Phi_G(Q,V) = E(Q) - QV \tag{6.13}$$

where $E(Q)$ is the total system energy defined in equation 6.4, and V is our external gate voltage. Using this expression, we now minimize $\Phi_G(Q,V)$:

$$\left.\frac{\partial \Phi_G(Q,V)}{\partial Q}\right|_{Q=Q_{eq}} = 0 \tag{6.14}$$

where we evaluate this derivative at each of our chosen Q values. Given that we have an analytic expression for $E(Q)$, we can easily take it's derivative to get:

$$\left.\frac{\partial \Phi_G(Q,V)}{\partial Q}\right|_{Q=Q_{eq}} = a_0 + \left(a_1 + \frac{z_0 - z_{ref}}{\epsilon_0 A}\right) Q + \frac{Q}{C} - W = V \tag{6.15}$$

which now allows us to write the grand potential as a function of voltage:

$$\Phi_G(V) = E(Q_{eq}(V)) - Q_{eq}(V)V \tag{6.16}$$

In practice, it is necessary to interpolate the results obtained from equation 6.16 for each phase so that $\Phi_G(V)$ can be calculated using the same voltages for both *Pnma* and *Cmcm*. The transition voltage is then obtained by finding the intersection of the curves generated using the interpolated grand potential functions.

6.3 Phase Transition via Mechanical Strain

In addition to electrostatic gating, we were also interested in how mechanical strain can be used to drive a structural phase transition in 2D SnS. Previous work has been done studying these effects in monolayer TMD materials, which showed promising results (Duerloo et al., 2014).

6.3.1 Strain Driven Transition of SnS

To perform these calculations, a python script was set up for each phase. In both cases, the initial lattice constants were chosen to match those of structurally relaxed *Pnma*, though the atom positions used for the *Cmcm* phase were obtained from a separate relaxation where the correct *Cmcm* lattice constants were used. A percentage strain was applied to the lattice constants of each phase, after which the structure was relaxed until a convergence criteria of 1×10^{-5} eV was met, though only the atom positions where allowed to move, as we wished to keep the stressed lattice constants fixed[6]. Once the relaxation was complete, a final energy calculation was performed using the atom positions from the previous relaxation. Both the relaxation and energy calculations were done using a 16x16x1 Monkhorst-Pack k-point mesh, with an energy cutoff of 600 eV (Monkhorst and Pack, 1976). Gaussian smearing was used for the relaxations, while the tetrahedron method with Blöchl corrections[7] was used for energy calculations. A smearing width of 50 meV was used in both cases. The strain simulations were performed on a 10x10 grid, with values ranging from a 10% compression to 6% strain on the initial *Pnma* lattice constants. The calculated energies at each value were interpolated using the **interp2d** Scipy function, allowing us to create a finer grid of values.

When no stress has been applied to the system, the *Pnma* phase has a lower internal energy than the *Cmcm* phase. However, as the lattice constants (a, b) are changed due to the introduction of strain/compression, the internal energy of each phase is altered. For certain values of (a, b), the energy associated with the *Pnma* phase becomes larger than that of the *Cmcm* phase. In these cases, a phase transition will occur, as the system will prefer to be in the state with the lowest internal energy (Duerloo et al., 2014). To determine the stresses that are required to drive this transition, we generate a contour plot showing the difference in energy between the two phases as stresses are applied to the initial *Pnma* lattice constants (a_0, b_0). The contour where the difference between the energies is zero describes the range of percentages by which the initial lattice constants would need to be changed in order for a transition to occur. Using this method, we predict that a phase transition will occur when the b axis is compressed by at least 6%.

While our calculations show that a 6% compression will be required for a transformation to occur, other works have shown that the inclusion of thermal corrections, as well as hybrid-DFT techniques can improve this value, moving it closer to equilibrium (Duerloo et al., 2014).

6.4 Temperature-induced Phase Transition

An alternative to inducing the phase transition in SnS monolayers via electrical doping or strain is to heat the monolayer. To predict a transition temperature, we need to be able to characterize the

[6] ISIF = 2
[7] ISMEAR=-5

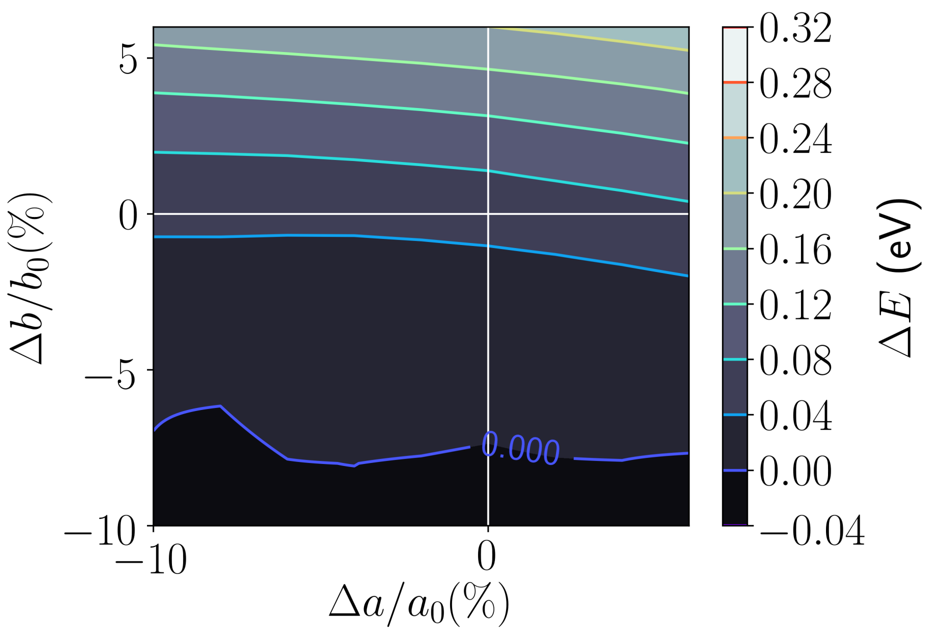


Figure 6.8: Contour plot showing how the energy difference between the phases changes as a function of the percent difference of the initial *Pnma* lattice constants (a_0, b_0).

thermodynamic properties of the *Cmcm* and *Pnma* phases by investigating the phonon modes of both phases. Phonons refer to the vibrational modes of a crystal lattice and can be used to calculate a basic estimate for the free energy, entropy, and heat capacity of a material given the phonon density of states (Dove, 1993).

In the harmonic approximation for calculating phonon modes, atoms in the crystal lattice are treated as behaving like springs in that they exhibit a restoring force back to their equilibrium positions. A set of force constants or "spring" constants are defined between each pair of atoms where the term $\phi_{i,j}$ refers to the force constant between the i^{th} and j^{th} atoms. Note that this approximation does not restrict the force constants to being equivalent to one another.

It is possible to calculate phonons using the harmonic approximation using either 1) the finite displacement method or 2) density functional perturbation theory. The harmonic approximation results presented in this report were calculated using the finite displacement method in which a set of systematic displacements based on the symmetry of the crystal is generated and then an electronic relaxation using DFT is performed on each distorted structure to find the force constants $\phi_{i,j}$. Given each force constant, an eigenvalue problem is solved to find the normal modes of the system, each with an angular frequency $\omega(\mathbf{k}, \nu)$ that is a function of the location in reciprocal space $\mathbf{k}$ and the band index ν. Each phonon mode $\omega(\mathbf{k}, \nu)$ corresponds to a set of displacements of the atoms in the real-space crystal determined by the mode eigenvectors (Togo and Tanaka, 2015). A heuristic illustration of this procedure is depicted in Figure 6.9 for a NaCl crystal.

To calculate the phonon dispersion curves of the two phases using DFT at 0 K, a 5x5x1 supercell of each phase was generated using the Python phonopy package (Togo and Tanaka, 2015). The finite displacements were generated with the default 0.01 Å displacement size and `DIAG = .FALSE.`,

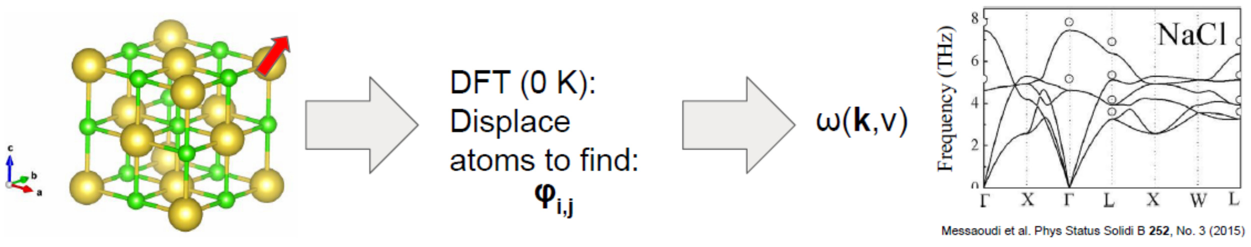


Figure 6.9: Heuristic illustration of the finite displacement method for calculating phonon modes $\omega(\mathbf{k},\nu)$ using DFT at 0 K.

so that displacements in the diagonal direction are not generated.[8] Then a set of electronic relaxations were calculated on the distorted structures using VASP and the PBE functional with an energy cutoff of 400 eV, a convergence criterion of 1×10^{-8} eV, and a Monkhorst-Pack k mesh of $3\times3\times1$. Using the `vasprun.xml` output files from VASP, the phonon dispersion curves and density of states were calculated in phonopy using a q mesh of $64\times64\times64$ for the frequencies -2.06 THz to 9.88 THz in steps of 0.01 THz and `FC_SYMMETRY=.TRUE.`. To estimate a transition temperature from the 0 K calculations, phonopy was used to calculate the phonon contribution to the Helmholtz free energy from the phonon density of states and the ground state energy of each supercell was calculated using VASP and added to the corresponding phase's phonon free energy.

6.4.1 Displacive Phase Transition of SnS

The dispersion curve calculated for the high energy *Cmcm* phase is shown in Figure 6.10. It exhibits the expected anharmonicities, indicating that the *Cmcm* structure is unstable at 0 K. Each anharmonicity corresponds to an imaginary mode that is solved from the dynamical matrix (eigenvalue problem). An imaginary mode frequency implies that upon distortion by the mode eigenvector, the initial structure will continue to deform under that mode's strain rather than exhibit a restoring force back to its equilibrium position, thus deforming to a lower energy structure (Dove, 1993).

The calculated *Cmcm* dispersion indicates that there are two anharmonic modes at the Γ point. By inspecting these modes in VESTA (Momma and Izumi, 2008), it is clear that they are both shear modes in the a-b plane. This shearing corresponds to the transformation of *Cmcm* into one of the two instances of *Pnma* upon cooling. Since the *Cmcm* phase corresponds to a thermal average of the two instances of *Pnma* (see Figure 6.1) we expect there to be two of these degenerate shear modes, as verified by the calculated phonon dispersion at 0 K shown in Figure 6.10.

Having confirmed that temperature can be used to induce a phase transition between the *Cmcm* and *Pnma* phases, our next goal was to estimate this transition temperature. Using the density of states $g(\omega)$ calculated from the 0 K dispersions curves, the transition temperature was estimated to occur above 2400 K. However, since the bulk transition temperature has been measured to be around 875 K (Skelton et al., 2017) and since the monolayer transition temperature should be around

[8]Phonopy recommends setting `DIAG` to `.FALSE.` for unit cells with one dimension much longer than the other two, e.g. a monolayer with vacuum space.

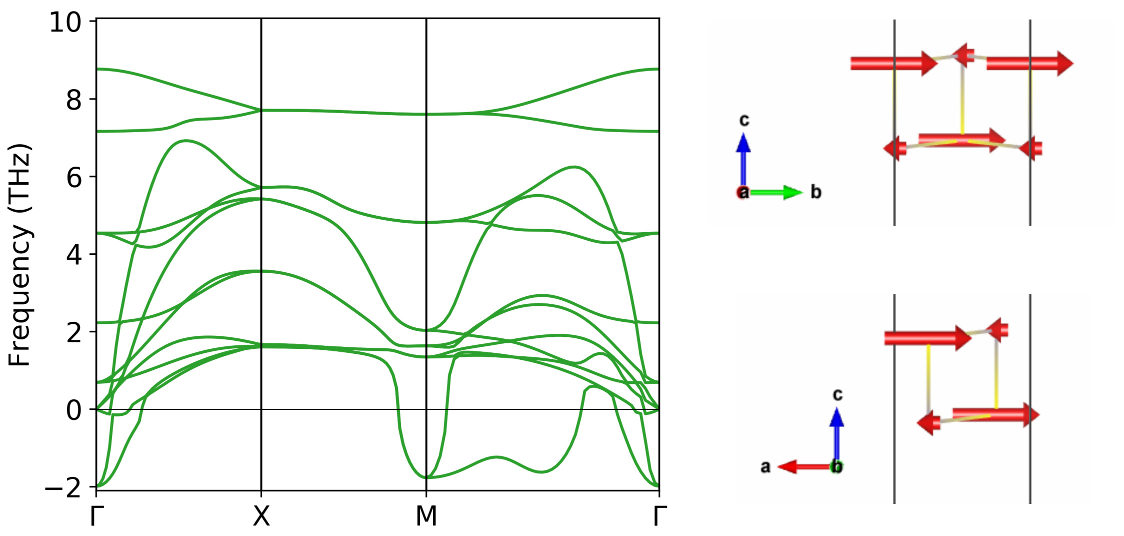


Figure 6.10: Left: Phonon dispersion curve of the *Cmcm* monolayer calculating using the finite displacement method at 0K using the PBE functional and phonopy package. The imaginary, shear modes at Γ, shown as negative frequencies in this plot, indicate a displacive phase transition to the *Pnma* phase upon cooling. Right: The two degenerate shear modes at $2i$ THz at Γ. Both modes displace the Sn and S atoms in the a-b plane, indicating a shearing of the *Cmcm* phase to one of the two instances of the *Pnma* phase.

that temperature or lower, this estimate, along with the anharmonicites evident in the *Cmcm* 0 K dispersion curve, indicated that it was necessary to account for the temperature dependence of the phonon mode frequencies, motivating the use of molecular dynamics simulations.

6.4.2 Incorporating Phonon Temperature Dependence

To incorporate the phonon frequency temperature dependence, a new post-processing package Temperature Dependent Effective Potential Method (TDEP) was used to calculate the phonon dispersion curves using output from molecular dynamics (MD) simulations (Hellman et al., 2013). MD simulations are similar to the finite displacement method used in the DFT 0 K calculations, however, rather than displacing the atoms based on inherent symmetries in the crystal, an external thermostat is used to displace the atoms randomly by thermal energy in the system. Once these atoms are displaced, a set of forces are calculated, the atoms are displaced in time based on the forces, and the process is repeated.

Given an MD simulation over N_t time steps, TDEP calculates the harmonic force constants between pairs of atoms by mapping the MD-calculated forces $\boldsymbol{F}_t^{MD}$ and displacements $\boldsymbol{U}_t^{MD}$ at each time step t onto a harmonic model. The TDEP program employs a least squares mapping at each time step which minimizes the differences between the simulated (MD) forces $\boldsymbol{F}_t^{MD}$ and the harmonic force $\boldsymbol{F}_t^H = \Phi \boldsymbol{U}_t^{MD}$ via in Equation 6.17, where Φ is the matrix of unknown force

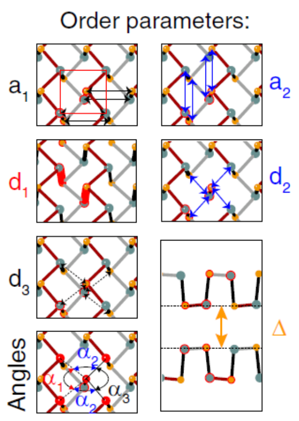


Figure 6.11: Lattice order parameters for the *Pnma*phase from (Mehboudi et al., 2016).

constants:

$$\min_{\boldsymbol{\Phi}} \Delta \boldsymbol{F} = \frac{1}{N_t} \sum_{t=1}^{N_t} |\boldsymbol{F}_t^{MD} - \boldsymbol{F}_t^{H}|^2 \tag{6.17}$$

The solution, Φ, to the least square mapping is given by:

$$\boldsymbol{\Phi} = \left[\boldsymbol{F}_1^{MD}\, \boldsymbol{F}_2^{MD} ... \boldsymbol{F}_{N_t}^{MD}\right] \left[\boldsymbol{U}_1^{MD}\, \boldsymbol{U}_2^{MD} ... \boldsymbol{U}_{N_t}^{MD}\right]^+ , \tag{6.18}$$

where $^+$ refers to the Moore-Penrose pseudoinverse (Hellman et al., 2013).

Molecular dynamics calculations were performed using VASP with a Nosé-Hoover thermostat and Fermi smearing with `SIGMA` set to $k_B T$, where T is the temperature of the thermostat. A time step of 2 fs was used as well as an energy cutoff of 400 eV and a convergence criterion of $1{\times}10^{-4}$ eV. In order to balance computational speed and accuracy, `PREC` was set to `Single` and `ALGO` was set to `VeryFast`. An 8×8×1 supercell was generated with TDEP using the command `generate_structure -d 8 8 1`. Both the *Pnma* and *Cmcm* structures were simulated at 200 K, 300 K, 400 K, 500 K, and 800 K for 10,000 time steps each. Additionally, eight lattice order paramters were monitored at each time steps using an `ICONST` file; a_1, a_2, d_1, d_2, d_3, correspond to interatomic distances while α_1, α_2, and α_3 correspond to interatomic angles. These order parameters are illustrated for the *Pnma* phase in Figure 6.11 from (Mehboudi et al., 2016). Alternatively, since the `ICONST` file was incorrect for the first 4,900 of 10,000 steps for the *Pnma* run at 200K, an additional script, `getLatParams.py`, was developed that can extract these order parameters for a specified set of atoms from the `XDATCAR` atom positions after the VASP run has completed.

Using the MD-calculated `OUTCAR` files, TDEP was used to calculate the second order force constants and extract the phonon dispersion curves and density of states at each temperature for each phase on a q mesh of 64×64×1 grid via the commands `extract_forceconstants -rc2 5` and `phonon_dispersion_relations -rp -qg 64 64 1 --dos`, for the appropriate monolayer reciprocal space path.

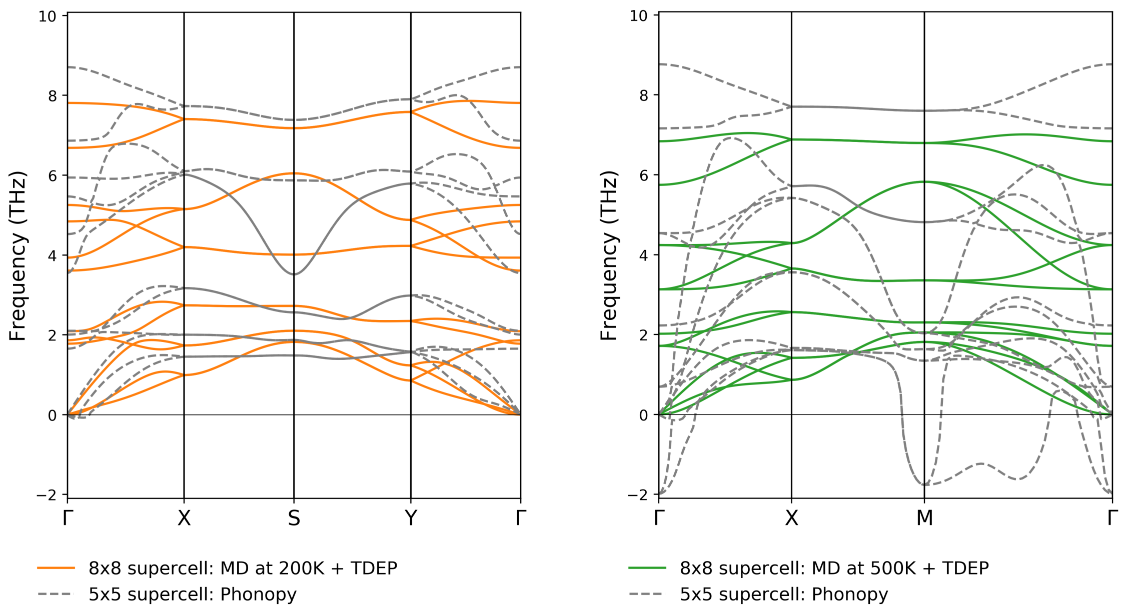


Figure 6.12: Left: Calculated dispersion curves for the *Pnma* phase of monolayer SnS. Right: Calculated dispersion curves for the *Cmcm* phase of monolayer SnS.

6.4.3 Dispersions of the Two Phases

The dispersions and density of states calculated for the *Pnma* phase at 200 K and the *Cmcm* phase at 500 K from time steps 300 to 4800 are shown in Figure 6.12 as a comparison with the 0 K finite displacement results calculated using phonopy.

Notably, the anharmonic modes at the Γ and M points in the 0 K calculated *Cmcm* dispersion disappear for the TDEP calculated dispersion which was extracted from MD simulations made at 500 K. Thus the TDEP method is able to capture the temperature dependence of the phonon modes by sampling the Born-oppenheimer potential energy surface (Hellman et al., 2013).

In addition, Figure 6.12 demonstrates that overall, the TDEP calculated dispersions are lower in energy than the 0 K calculated dispersions. This discrepancy is not entirely unexpected as the potential energy surfaces being sampled differ between the two methods, with one surface neglecting temperature dependence.

6.4.4 Estimating a Transition Temperature

In order to obtain a “ball-park” estimate for the transition temperature, we calculated the Helmholtz free energy $F(T)$ as a function of temperature from 200 K to 400 K using the density of states extracted for the *Pnma* phase at 200 K and *Cmcm* at 500 K using TDEP. Note that we assumed that we could neglect the temperature dependence of the density of states in this ball-park estimate. However, given the additional MD simulations at other temperatures, it would be useful in future work to calculate the temperature dependence of this density of states and obtain an estimate for the uncertainty in this transition temperature.

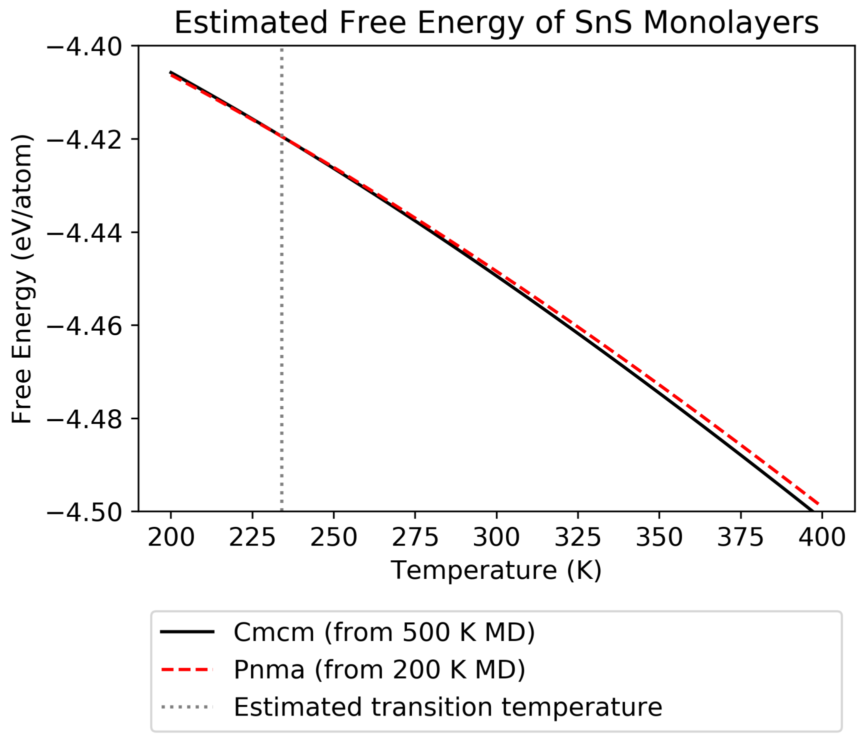


Figure 6.13: Estimated free energies of the *Cmcm* and *Pnma* phases as a function of temperature calculated using the TDEP density of states at 500 K and 200 K respectively. A transition temperature of approximately 240 K is indicated by the dotted vertical line.

The temperature dependent phonon free energy $F_{vib}(T)$ is calculated using the TDEP command `phonon_dispersion_relations -temperature-range 200 400 201`. The ground state energy, U_0, is calculated as the average difference between the MD-calculated ground state energy (energy without entropy) at each time step and the ground state energy due to the harmonic force constants using the TDEP command `extract_force_constants -rc2 5 --U0`. The final free energy is then given by (Hellman et al., 2013):

$$F(T) = U_0 + F_{vib}(T). \tag{6.19}$$

Figure 6.13 depicts the predicted free energy of the *Cmcm* phase (solid black line) and the *Pnma* phase (dashed red line) as a function of temperature. Below the estimated transition temperature of 240 K, *Cmcm* is the high energy phase. Above the transition temperature, *Pnma* is the high energy phase. These results confirm our expectation that the SnS monolayer exhibits a displacive phase transition between its two structural phases that can be induced via temperature.

It is important to note that this transition occurs well below room temperature. A useful experimental verification would be to determine whether SnS, in its monolayer state, is in the *Cmcm* or *Pnma* phase at room temperature. This may be difficult in practice, however, as SnS has so far only been synthesized using thin-films and not single monolayers (Higashitarumizu et al., 2018). On the computational side, it will be beneficial to verify this transition by monitoring the lattice order parameters near the transition temperature, assuming we can generate enough time steps to observe the transition kinetics. Alternatively, as suggested above, it will be imperative to incorporate the temperature dependence of the phonon density of states to calculate the transition temperature and/or the uncertainty in this “ball-park” estimate.

6.4.5 Monitoring the Lattice Order Parameters

As a means of verifying the structural transition of SnS above the estimated 240 K transition temperature, several lattice order parameters were monitored at each time step in the MD simulations. As shown in Figure 6.11, the lattice order parameters correspond to five interatomic distances and three interatomic angles.

A python script, `getLatParams.py`, was written so that these lattice order parameters could be extracted from the atom positions in the `XDATCAR` files for each MD run of the $8\times8\times1$ supercells generated using TDEP. Figure 6.14 depicts the five interatomic distances monitored as a function of time for the two different phases at a temperature of 200 K and 500 K, based on the average of approximately 40 different pairs of atoms monitored. All pairs of atoms were taken from the uppermost c-level of the supercell, but additional atoms from the lower c-level could be added to the code in the future if more pairs of atoms are needed. Compared to plotting the trajectory of an interatomic distance between a single pair of atoms, the average over 40 pairs exhibits much smaller fluctuations in time.

The *Pnma* phase was simulated at a temperature of 200 K, well below the estimated transition temperature of 240 K. As shown in Figure 6.14a), all of the interatomic distances remain distinct from one another throughout all time steps. In particular, the inter-zizag distance a_1 remains approximately 0.2 Å greater than the intra-zizag distance a_2, and d_3 which describes the distance between atoms in the c-level above a bonded pair remains approximately 0.5 Å greater than the

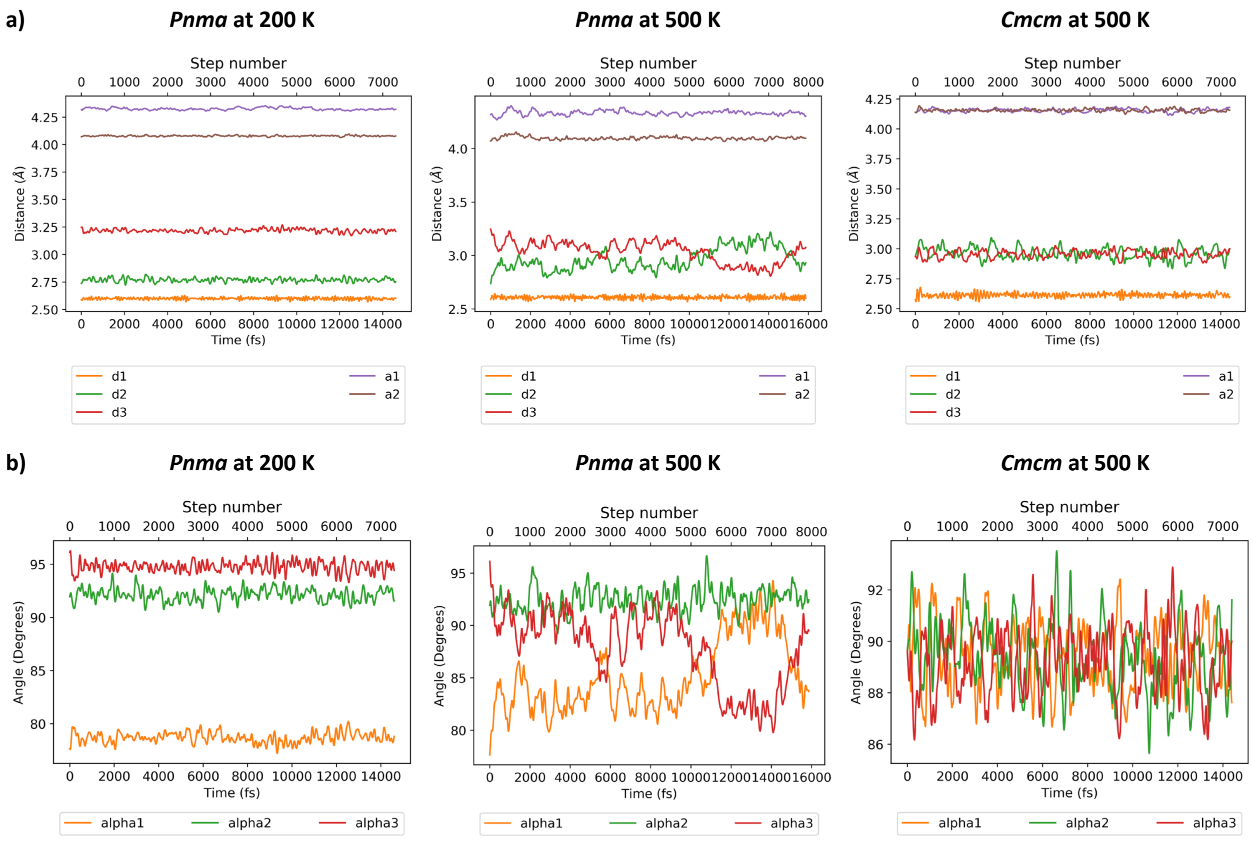
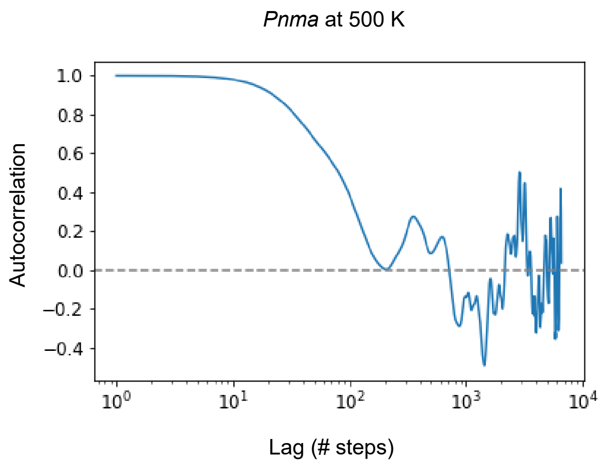


Figure 6.14: Lattice order parameters as a function of time from the molecular dynamics simulations for a) interatomic distances and b) interatomic angles. The *Pnma* to *Cmcm* transition is apparent at 500 K as the order parameters d_2 and d_3, and α_1 and α_3 start to fluctuate around one another at 6000 fs and after but not in the order parameters a_1 and a_2 which remain distinct from one another by approximately 0.2 Å for all time steps shown.



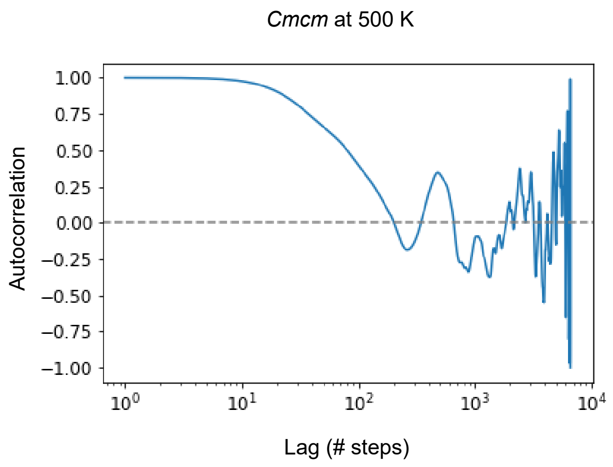


Figure 6.15: Time auto-correlation of the lattice parameter a_1 as a function of the lag for *Pnma* and *Cmcm* at 500 K. The auto-correlation was calculated from the molecular dynamics output after 1,200 fs and up to 14,400 fs for *Pnma* and 15,864 fs for *Cmcm*. For both phases, a lag of 200 time steps, or 400 fs, exhibits minimal (~0.01 to 0.02) correlation.

distance between a bonded pair of atoms on a single c-level, d_2. Similarly, in Figure 6.14b), all of the interatomic angles α_1, α_2, and α_3 remain unequal through all time. In contrast, for the *Cmcm* phase simulated at a predicted stable temperature of 500 K as shown in Figure 6.14a), the distances a_1 and a_2, and d_3 and d_2 are on average equal to one another across all time steps. In addition, all of the interatomic angles in the *Cmcm* phase oscillate about approximately 90 degrees.

The *Pnma* phase was then simulated at 500 K, well above the expected transition temperature. A transition is notable in the d_3 and d_2 parameters, as the two start to oscillate about an average value of 3.00 Å starting after 6000 fs has elapsed, and in the oscillation of the interatomic angles α_1 and α_3. This trend of oscillation in the d_2 and d_3 parameters and interatomic angles has been observed in the monitoring of other monolayer monochalcogenides above the transition temperature in (Mehboudi et al., 2016). However, within 16,000 fs, a transition is not evident in the a_1 and a_2 parameters. It is possible that more time steps are required to see the transition in a_1 and a_2, since MD simulations for phase transitions can often take thousands to hundreds of thousands of time steps. In addition, it is possible that the lattice is too constrained for the transition to be observable using MD trajectories; one alternative would be to strain the lattice constants towards the *Cmcm* phase and interpolate on either side of the transition temperature. Finally, along a similar vein, it is possible that the a_1 and a_2 parameters are more a measure of the lattice constants of the overall supercell, which are constrained to the starting cell configuration, than a metric for the overall phase of the supercell.

Autocorrelation of Lattice Parameters

In order to understand the number of time steps required to sample from the output of a molecular dynamics run, we computed the time auto-correlation of the monitored lattice parameters for the two phases of SnS at 500 K as a function of lag. Here, lag refers to the number of time steps between samples picked from a molecular dynamics run to compute statistical averages. In order to avoid large fluctuations due to equilibration early in the molecular dynamics run, the auto-correlation for each phase and parameter was computed after 600 time steps, or 1,200 fs.

As demonstrated in Figure 6.15, both *Pnma* and *Cmcm* exhibit minimal auto-correlation ($\sim$0.01-0.02) for a lag of 200 time steps or 400 fs. Correlation is completely lost for lags of 2,000 time steps or more. However given that we have a maximum of 10,000 time steps simulated for each phase at each temperature, these results suggest that the minimum uncorrelated sampling required is time steps 600 to 10,000, in steps of 200, corresponding to a sample size of just under 50 time steps.

6.5 Conclusion

Here, we completed DFT simulations to investigate the properties of structural phase transitions in monolayer SnS driven by electrostatic gating, mechanical strain, and temperature. Through the application of gating, we predict a phase transition to occur in SnS when a transition voltage of -8.78 V is applied. Additionally, this method was applied to monolayer GeSe, where similar results were seen, with an expected voltage of -6.75 V being necessary to drive the transition. The modification of the monolayer SnS lattice constants through mechanical strain was also simulated, where it was found that a compression greater than 6% is expected to transform the structure from the *Pnma* to *Cmcm* phase.

Based on the lattice dynamics work presented in this report, we do expect a displacive phase transition from the *Cmcm* to the *Pnma* phase of an SnS monolayer upon cooling. We conducted molecular dynamics simulations and processed the results using TDEP to calculate phonon dispersions and density of states which incorporate the temperature dependence and large anharmonicites of the high temperature phase. A coarse estimate based on the phonon density of states of the *Cmcm* phase at 500 K and the *Pnma* phase at 200 K predicts a transition temperature in the vicinity of 240 K, below room temperature.

6.5.1 Future Work

In the future, we plan to conduct a more thorough analysis of the transition temperature based on changes in the phonon density of states with temperature. In addition, we plan to use correlation statistics to analyze the average lattice order parameters at different temperatures as a means of verifying the transition temperature. Furthermore, we would like to investigate applications of the thermally-induced structural phase transition including thermal conductivity and phonon-phonon scattering for each of the two phases.

Finally, the addition of thermal corrections to strain and gating simulations has been shown to lower the strain and voltage needed to drive the phase transition in other 2D materials (Duerloo et al., 2014; Rehn et al., 2018a). Application of these same methods to monolayer SnS and other 2D group IV monochalcogenides would be useful in improving our understanding of these transition properties.

Quinn M. White *is a senior at Arizona State University majoring in Physics and Mathematics.*

His usual research is focused on using quantum Monte Carlo methods to study the properties of neutron star matter. Upon completion of his undergraduate degrees he intends to pursue a PhD in theoretical condensed matter or high energy physics.

***EliseAnne C. Koskelo** is an incoming graduate student at the University of Cambridge where she plans to study frustrated magnetism in 2D materials and Ising garnets. She graduated from Pomona College this spring with a double-major in Physics and Mathematics. Her theses research was on "stochastic resonance," using noise to enhance the resolution of thermoreflectance imaging.*

Acknowledgements

We would like to thank our mentors, Daniel Sheppard, Daniel Rehn, and Ann Mattson for their helpful guidance and support throughout this research project, and the workshop organizers, Dan Israel, Madison Andrews, and Garry Maskaly for making this wonderful summer experience possible. We also would like to thank Evan Reed for helpful discussions.

Bibliography

Ann Mattson. DFT for DFT Users, July 2020. LA-UR-20-25346.

P. E. Blöchl. Projector augmented-wave method. *Phys. Rev. B*, 50:17953–17979, Dec 1994. doi: 10.1103/PhysRevB.50.17953. URL `https://link.aps.org/doi/10.1103/PhysRevB.50.17953`.

Klaus Capelle. A bird's-eye view of density-functional theory. *arXiv:cond-mat/0211443*, November 2006. URL `http://arxiv.org/abs/cond-mat/0211443`. arXiv: cond-mat/0211443.

D. M. Ceperley and B. J. Alder. Ground state of the electron gas by a stochastic method. *Phys. Rev. Lett.*, 45:566–569, Aug 1980. doi: 10.1103/PhysRevLett.45.566. URL `https://link.aps.org/doi/10.1103/PhysRevLett.45.566`.

Martin T. Dove. *Introduction to Lattice Dynamics*. Cambridge Topics in Mineral Physics and Chemistry. Cambridge University Press, 1993. doi: 10.1017/CBO9780511619885.

Karel-Alexander N. Duerloo, Yao Li, and Evan J. Reed. Structural phase transitions in two-dimensional mo- and w-dichalcogenide monolayers. *Nature Communications*, 5, 2014. doi: 10.1038/ncomms5214.

Olle Hellman, Peter Steneteg, I.A. Abrikosov, and S. I. Simak. Temperature dependent effective potential method for accurate free energy calculations of solids. *Physical Review B*, 87:104111, 2013.

Naoki Higashitarumizu, Hayami Kawamoto, Masaru Nakamura, Kiyoshi Shimamura, Naoki Ohashi, Keiji Ueno, and Kosuke Nagashio. Self-passivated ultra-thin sns layers via mechanical exfoliation and post-oxidation. *Nanoscale*, 10:22474, 2018.

P. Hohenberg and W. Kohn. Inhomogeneous electron gas. *Phys. Rev.*, 136:B864–B871, Nov 1964. doi: 10.1103/PhysRev.136.B864. URL `https://link.aps.org/doi/10.1103/PhysRev.136.B864`.

G. Kresse and J. Furthmüller. Efficient iterative schemes for ab initio total-energy calculations using a plane-wave basis set. *Phys. Rev. B*, 54:11169–11186, Oct 1996. doi: 10.1103/PhysRevB.54.11169. URL `https://link.aps.org/doi/10.1103/PhysRevB.54.11169`.

Yao Li, Karel-Alexander N. Duerloo, Kerry Wauson, and Evan J. Reed. Structural semiconductor-to-semimetal phase transition in two-dimensional materials induced by electrostatic gating. 7.

Mershad Mehboudi, Benjamin Fregoso, Yurong Yang, Wenjuan Zhu, Arend van der Zande, Jaime Ferrer, L. Bellaiche, Pradeep Kumar, and Salvador Barraza-Lopez. Structural phase transition and material properties of few-layer monochalcogenides. *Physical Review Letters*, 117:246802, 2016.

Koichi Momma and Fujio Izumi. Vesta: a three-dimensional visualization system for electronic and structural analysis. *Journal of Applied Crystallography*, 41(3):653–658, 2008. doi: 10.1107/S0021889808012016. URL `https://onlinelibrary.wiley.com/doi/abs/10.1107/S0021889808012016`.

Hendrik J. Monkhorst and James D. Pack. Special points for brillouin-zone integrations. *Phys. Rev. B*, 13:5188–5192, Jun 1976. doi: 10.1103/PhysRevB.13.5188. URL `https://link.aps.org/doi/10.1103/PhysRevB.13.5188`.

J. P. Perdew and Alex Zunger. Self-interaction correction to density-functional approximations for many-electron systems. *Phys. Rev. B*, 23:5048–5079, May 1981. doi: 10.1103/PhysRevB.23.5048. URL `https://link.aps.org/doi/10.1103/PhysRevB.23.5048`.

John P. Perdew, Kieron Burke, and Matthias Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, 77:3865–3868, Oct 1996. doi: 10.1103/PhysRevLett.77.3865. URL `https://link.aps.org/doi/10.1103/PhysRevLett.77.3865`.

Daniel A. Rehn, Yao Li, Eric Pop, and Evan J. Reed. Theoretical potential for low energy consumption phase change memory utilizing electrostatically-induced structural phase transitions in 2D materials. *npj Computational Materials*, 4(1):1–9, January 2018a. ISSN 2057-3960. doi: 10.1038/s41524-017-0059-2. URL `https://www.nature.com/articles/s41524-017-0059-2`.

Daniel A Rehn, Yao Li, and Evan J Reed. Refrigeration in 2d: Electrostaticaloric effect in monolayer materials. *Physical Review Materials*, 2(11):114004, 2018b.

Johnathan M. Skelton, Lee A. Burton, Fumiyashi Oba, and Aron Walsh. Chemical and lattice stability of the tin sulfides. *The Journal of Physical Chemistry C*, 121:6446–6454, 2017.

Jonathan M. Skelton, Lee A. Burton, Stephen C. Parker, Aron Walsh, Chang-Eun Kim, Aloysius Soon, John Buckeridge, Alexey A. Sokol, C. Richard A. Catlow, Atsushi Togo, and Isao Tanaka.

Anharmonicity in the high-temperature $cmcm$ phase of snse: Soft modes and three-phonon interactions. *Phys. Rev. Lett.*, 117:075502, Aug 2016. doi: 10.1103/PhysRevLett.117.075502. URL `https://link.aps.org/doi/10.1103/PhysRevLett.117.075502`.

Atsushi Togo and Isao Tanaka. First principles phonon calculations in materials science. *Scripta Materialia*, 108:1 – 5, 2015. ISSN 1359-6462. doi: https://doi.org/10.1016/j.scriptamat.2015.07.021. URL `http://www.sciencedirect.com/science/article/pii/S1359646215003127`.

Ying Wang, Jun Xiao, Hanyu Zhu, Yao Li, Yousif Alsaid, King Yan Fong, Yao Zhou, Siqi Wang, Wu Shi, Yuan Wang, Alex Zettl, Evan J. Reed, and Xiang Zhang. Structural phase transition in monolayer $MoTe_2$ driven by electrostatic doping. *Nature*, 550, 2017. doi: 10.1038/nature24043.

Dante Zakhidov, Daniel A Rehn, Evan J Reed, and Alberto Salleo. Reversible electrochemical phase change in monolayer to bulk-like mote2 by ionic liquid gating. *ACS nano*, 14(3):2894–2903, 2020.

Chapter 7

Uncertainty Quantification in High Explosive Equations of State

Team Members
Nga Ying Lo and Brian W. Bell

Mentors
Jeffrey A. Leiding (T-1), Stephen A. Andrews (XCP-8) & Chris Ticknor (T-1)

Abstract

In this work, we performed ab initio molecular dynamic simulation (AIMD) of DAAF molecules in the product state to understand the behavior of the high explosive system at equilibrium and developed algorithms to identify the bounds of vapor domes – pressure-temperature-volume regimes where multiple chemical phases coexist. From AIMD, we obtained the average energies and total pressure of nuclear configurations sampled from equilibrium and a comparison with Magpie's total pressure revealed consistency between the two outputs from 4,000 K to 5,000 K. A radial distribution analysis was also performed to study the equilibrium composition at temperatures from 4,000 K to 10,000 K, and we observed a higher probability of N-N bond formation throughout the studied temperature regime (as compared to many other bond types). Meanwhile, the algorithms for computation of critical points are tuned off of unphysical non-monotonicity in modeled Gibbs Free energy and Pressure-Volume curves. The result is a set of volume bounds and a pressure that can be used to characterize the vapor dome and help correct the model's output for mixed chemical phases.

7.1 Introduction

Thermochemical codes are often used at LANL to study and predict the behavior of molecular fluid systems. However, understanding those of high explosives (HE) is much more challenging because the modeling spans over a wide range of *(P,V,T)* states. From previous work, we know that equations of state (EOS) describing chemical equilibrium has had some success in describing HE systems also because it accounts for the evolving of the system's conditions. Also, the products mixtures of

HE at chemical equilibrium can be modelled successfully using statistical mechanical perturbation theory.Ross (1979) These achievements thus motivate the development of a new thermochemical code called Magpie, which uses ideal mixing for modelling the mixing of the constituents by assuming that the interactions between all chemical species are similar such that the mixing enthalpy will be negligible.

Using Magpie, we are able to re-create HE product mixtures and return calculations of detonation properties, such as the pressure and density at the Chapman-Jouguet state, for different materials. Thus far, Magpie's predictions have been compared with experiments and other standard thermochemical codes such as CheetahTicknor et al. (2020), but validations using first principle techniques are also desired in order to understand properties of HE systems that cannot be measured in experiments, such as the pressure and energies distributions for states that are off of the Hugoniot, as well as the chemical composition. Through simulations, statistics of these properties can be collected and used to infer the molecular composition at equilibrium. Given the time constraints on this research, we have chosen molecular dynamics to simulate the molecular interactions of DAAF, a high explosive material developed at LANL.

In Magpie, also, modeling certain materials below their critical temperatures requires that we reconcile the behavior of mixed states under vapor dome, a region where current models exhibit un-physical behavior and will affect the hydrodynamic flow in HE mixture systems. To account for the existence of coexisting phase, algorithms must be developed to properly determine critical temperatures at which coexisting phase occurs.

For the rest of the report, we will provide the motivations for each project, and discuss the methods and results from these work separately followed by an overall conclusion.

7.1.1 AIMD

LANL is interested in a variety of HE formulations and many experiments were performed to study the behavior of historic HE. Since there is a lack of experimental data on relatively new HEs, we wished to benchmark Magpie's modeling using first principles information instead. DAAF, short for diaminoazoxyfurazan, is a new HE molecule developed in the late 1990s at LANL that we are interested in due to its desirable safety characteristics.Koch (2016)

From the information obtained by solving the electronic Schrödinger equation for an initial nuclear configuration, we can use ab initio molecular dynamics (AIMD) or ab initio reactive Monte Carlo (AIRxMC) simulations to propagate the system to a new a state thus sampling equilibrium. AIMD is a purely deterministic simulation technique in which the trajectories of particles are determined by numerically integrating Newton's equations of motion for all degrees of freedom. On the contrary, AIRxMC takes into consideration of stochasticity which allows for broad chemical sampling by performing reactive move types. Although AIRxMC is desirable due to its ability to quickly equilibrate chemistry, we opted to perform an AIMD study and focus on higher temperature regimes, where chemical reaction rates are faster, given the time limitation of this summer workshop.

7.1.2 Vapor Dome Computation & Correction

A primary use case within Magpie is to model a given material with a single material Ross fluid-perturbation type model.Ross (1979) This particular model does not account for multiple coexisting phases. This results in un-physical outputs for P-T combinations at which multiple phases coexist. To correct this while retaining the advantages of the Ross model, it is desirable to identify the coexistence regime in sub-critical materials from model output, compute critical Temperature, and characterize the coexistence regime for sub-critical temperatures. We will compute these characterizations with methods based on a Maxwell construction Schroeder (1999) using Model outputs of Gibbs free energy, pressure, and specific volume.

7.2 AIMD methods & results

7.2.1 Methods

We employed finite-temperature density functional theory (DFT)Mermin (1965) with the PBE functionalPerdew et al. (1996) and the projector augmented wave (PAW) techniqueBlöchl (1994); Kresse and Joubert (1999) as implemented in the DFT software package VASPKresse and Hafner (1993, 1994); Kresse and Furthmüller (1996); Kresse and Furthmüller (1996). Molecular dynamics (MD)Allen and Tildesley (1987) simulation in the canonical (NVT) ensemble with a Nosé-Hoover thermostatNosé (1984); Hoover (1985) was performed, and our MD time step was 0.5 fs. Our nuclear configuration for production MD runs consists of 6 DAAF molecules which equates to 114 atoms.

Before performing production MD runs, we studied the convergence of the total free energy (F), the difference in free energy between two representative configurations (ΔF) at a given density and temperature, and the excess pressure (the part due to interactions in the system) as a function of kinetic energy cutoff of the plane-wave basis set. The quality of a DFT calculation is determined in part by the size of the basis set. VASP allows us to control this with the kinetic-energy cutoff parameter; the larger the cutoff, the larger the basis set; however, before we can perform a convergence study, we must obtain representative (of equilibrium) molecular configurations at a few thermodynamic states spanning or range of interest. Although for this work we focused on the isochore 1.0 g/cc, in future work we will perform MD and Monte Carlo studies over a range of density and temperature of 1.0 - 3.0 g/cc and 1,000 - 10,000 K. Therefore, we performed short NVT MD runs with a small but reasonable energy cutoff of 700 eV in order to quickly equilibrate the systems. We started with random atomic configurations (generated by random sequential addition) at the relatively high temperature of 10,000 K where the systems equilibrated quickly. We then annealed the systems, by continuing the simulations at 5,000 K until equilibrium was reached and repeating this short run at 1,000 K. We performed this process at 1.0 g/cc and 3.0 g/cc respectively. We report a representative example of our convergence study on one of these states (1.0 g/cc, 10,000 K) in Figure 7.1. For energy cutoffs greater than 1,000 eV, we see convergence in ΔF to better than 0.1 eV (or 1×10^{-3} eV/atom), and a fraction of a kilobar in the external pressure. We ultimately chose an energy cutoff of 1,000 eV for our production MD runs as it struck an acceptable balance between accuracy and computational cost.

Finally, we checked the variability of our results with respect to the number of atoms in our simulations. For a few states, we doubled the number of atoms from 114 to 228 and determined the averages and standard errors of two standard deviations for the specific energies and external pressure using Flyvbjerg analysis. This analysis estimates the statistical error in static quantity by grouping serially correlated data into blocks of optimal size so that it will return average values that represent uncorrelated data. With Flyvbjerg, we took into account the serial correlation to return realistic uncertainties for the calculated means.

For both the specific energies and external pressure, we see that there is an overlapping between the ranges of the two configurations at a density-temperature state of (1.0 g/cc, 10,000 K), as shown in Table.7.1. From this benchmark, we see that there is a negligible difference in the specific energies and pressure between the two systems consisting different numbers of atoms. Due to time constraint, we decided to conduct MD simulation using a system of 114 atoms for a density of 1.0 g/cc and a small range of temperatures (from 4,000 K to 10,000 K) in order to accumulate sufficient data for performance of statistical analysis. In the future, we will carry out simulations at more density-temperature states to collect more statistics on the equilibrium properties of the DAAF molecular system.

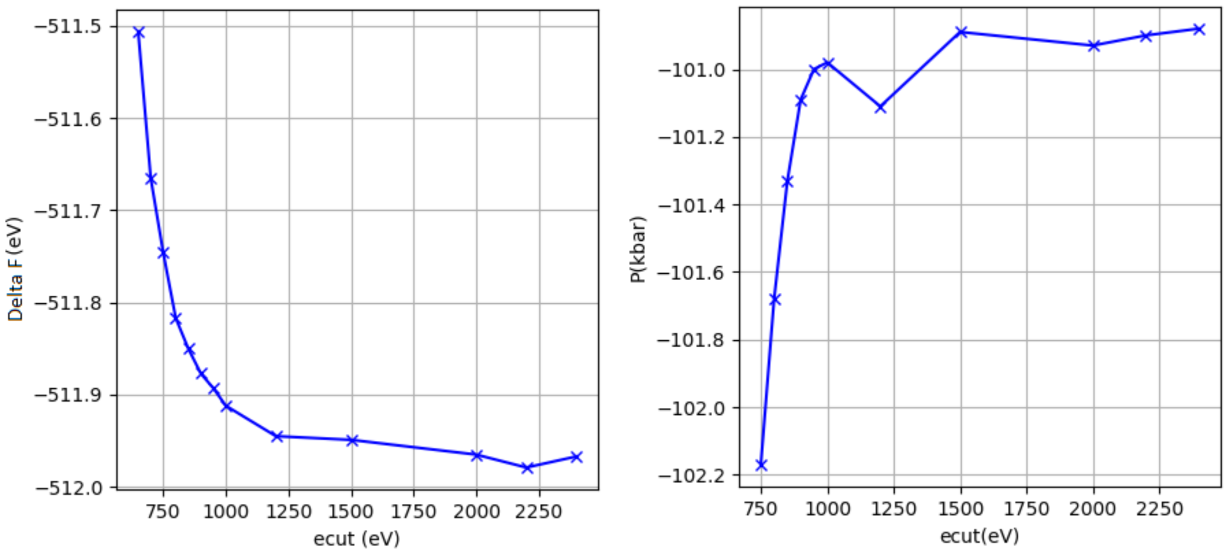


Figure 7.1: The convergence analysis of the external pressure and difference in free energy for a molecular system with a density of 1.0 g/cc at 10,000K. On the left is ΔF between two representative nuclear configurations; on the right is the P_{ext} for a given nuclear configuration. Both are plotted as a function of energy cutoff to determine the value at which the difference in free energy and pressure converge.

To determine the average energies and external pressures of the equilibrium nuclear configurations, Flyvbjerg analysis was performed on the data accumulated from approximately 5 to 10 picoseconds of MD simulation at each temperature: 4,000 K, 5,000 K, 6,000 K, 8,000 K and 10,000 K. Finally, the equilibrium composition was also studied using a radial distribution function analysis.

no. atoms	T (K)	$\langle F\rangle/atom$ (eV)	$\langle E0\rangle/atom$ (eV)	$\langle P_{ext}\rangle$ (kbar)
114	10000	(-6.014, -5.978)	(-5.670, -5.625)	(-6.646, -4.116)
228	10000	(-6.036, -5.999)	(-5.693, -5.648)	(-5.849,-3.761)

Table 7.1: Comparisons of external pressure and energies per atom for a system of 114 and 228 atoms with $(\rho, T) =$ (1.0 g/cc, 10000 K). The ranges presented represent the values of $\langle X\rangle \pm 2\sigma\,(\langle X\rangle)$.

7.2.2 Results

After the averages and errors were determined for the free energy, potential energy and the excess pressure from Flyvbjerg analysis, we also calculated for the ideal (kinetic) contributions to these quantities to understand how the total energy and pressure of the system evolve. The ideal pressure and energy were calculated using the ideal gas law and average kinetic energy equation for a monatomic gas:

$$p_{id}V = Nk_BT \tag{7.1a}$$

$$\langle E_k\rangle = \frac{3}{2}Nk_BT \tag{7.1b}$$

where k_B is the Boltzmann constant.

The results for a configuration of 114 atoms with a density of 1.0 g/cc are displayed in Tables.7.2-7.5 and plotted in Figures.7.2-7.5 as a function of temperatures. They are presented in the following order: free energy, sum of free and ideal energies, sum of potential and ideal energies, and the total pressure. From these figures, we observed that the energies and total pressure of the system increases with temperatures as expected. In addition to the total pressure values given from MD simulation, Table.7.5 and Figure.7.5 also provides the data and plot of the total pressure predicted by Magpie using statistical thermodynamics. From 4,000 K to 5,000 K, we observed a close agreement between the two data sets. Given a temperature of 4,000 K, the difference between Magpie's prediction of 3.989 GPa and AIMD's uppper limit of 3.933 GPa is only several hundredths of a GPa (0.056 GPa to be exact) even though Magpie's output does not fall exactly within the error-bar. This is a significant result because it indicates that Magpie's modeling of chemical equilibrium equations of state for high explosives is accurate for lower temperature ranges despite the fact that the thermochemical code does not take into account of any first principle techniques. At higher temperatures, however, we observed quite a significant deviation between the two outputs and this might be a result of additional physics processes that Magpie has no knowledge of. In the future, simulations at lower temperatures between 2,000 K to 4,000 K will be carried out for further benchmark study because that is the most important temperature regime for Magpie's modeling of HEs, as the detonation temperatures of most explosives are around 3,000 K.

T (K)	$\langle F \rangle$ (eV)	$\sigma_{\langle F \rangle}$ (eV)
4000	-758.32	0.62
5000	-742.23	0.73
6000	-728.81	0.97
8000	-710.19	1.35
10000	-685.97	2.15

Table 7.2: Mean and error on mean determined from Flyvbjerg analysis for the free energy at 1.0 g/cc.

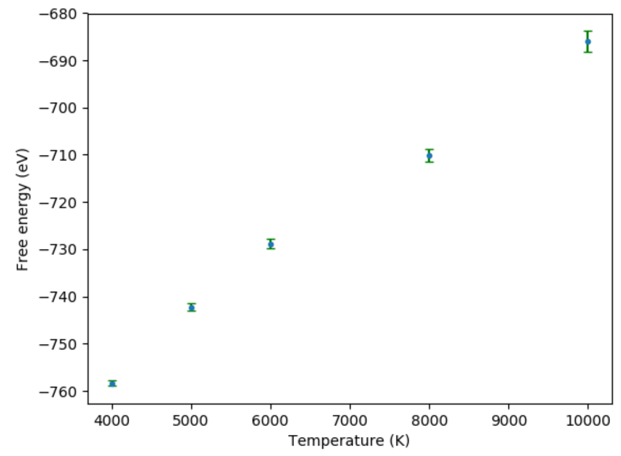


Figure 7.2: Plotted data shown in Table.7.2 for a system of 114 atoms at 1.0 g/cc.

T (K)	$\langle F \rangle + \langle E_k \rangle$ (eV)	σ (eV)
4000	-699.39	0.62
5000	-668.55	0.73
6000	-640.40	0.97
8000	-592.30	1.35
10000	-538.61	2.15

Table 7.3: Sum of Flyvbjerg calculated $\langle F \rangle$ and $\langle E_k \rangle$ along with the errors on $\langle F \rangle$ at different temperatures at 1.0 g/cc.

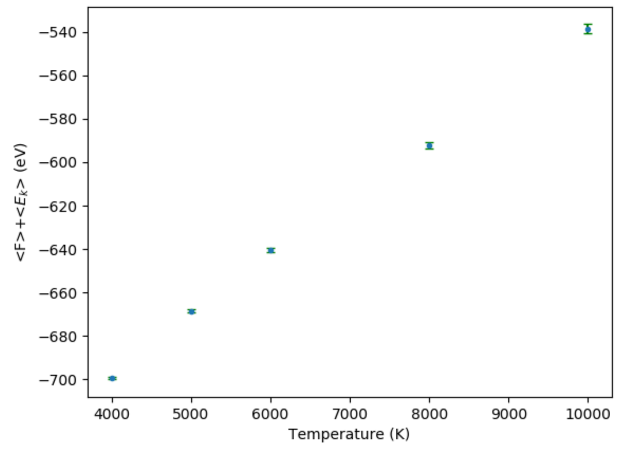


Figure 7.3: Plotted data shown in Table.7.3 as a function of temperature for a system of 114 atoms at 1.0 g/cc.

T (K)	$\langle E0 \rangle + \langle E_k \rangle$ (eV)	σ (eV)
4000	-696.83	0.64
5000	-663.57	0.77
6000	-631.90	1.11
8000	-573.57	1.61
10000	-499.05	2.59

Table 7.4: Sum of Flyvbjerg calculated $\langle E_0 \rangle$ (potential energy) and $\langle E_k \rangle$ along with the calculated errors on $\langle E_0 \rangle$ at different temperatures at 1.0 g/cc.

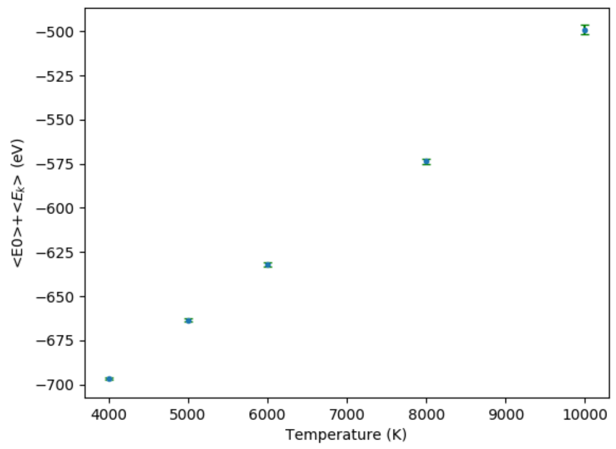


Figure 7.4: Plotted data shown in Table.7.4 as a function of temperature fora system of 114 atoms at 1.0 g/cc.

T (K)	Magpie $P_{tot}(GPa)$	MD P_{tot} (GPa)	MD σ (GPa)
4000	3.989	3.857	0.076
5000	4.752	4.557	0.105
6000	5.493	5.237	0.088
8000	7.118	6.493	0.105
10000	9.309	6.909	0.127

Table 7.5: The absolute pressure values of DAAF products at (1.0 g/cc) as predicted by Magpie and from MD simulation by determining the sum of Flyvbjerg calculated $\langle P_{ext} \rangle$ and the ideal pressure along with the error on the total pressure.

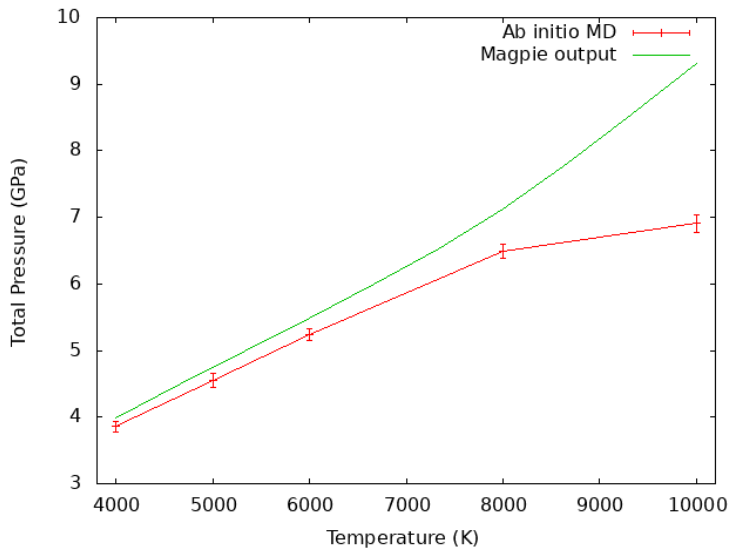


Figure 7.5: Plotted data shown in Table.7.5 for comparison of total pressure given by the MD simulation and Magpie in system of 114 atoms at 1.0 g/cc.

A radial distribution analysis was also performed to give us insight into the equilibrium composition of DAAF. From Figure.7.6, a selection of radial distribution functions *g(r)* are presented as a function of distance of separation between particles. Generally, there is a higher probability of chemical bonding at lower temperatures. For instance, looking at *g(r)* for N-N in Figure.7.2A, the peaks at approximately 1.1 Å for 4,000 K and 10,000 K are different by a factor ∼2.3. Signatures of other covalent bonds are seen between 1-1.5 Å in the other *g(r)* plots shown in Figure.7.6. At the highest temperature studied of 10,000 K, *g(r)* for N-N bond is at least 6.0 whilst other chemical bonds have a *g(r)* value between 1.5 and 3.0, which indicates that the abundance of molecules with a N-N bond is higher than those with other types of chemical bonds even at a higher temperature. This is expected since N-N has a very strong covalent bond and requires relatively large temperatures (and kinetic energies) to dissociate.

In Figure. 7.6B & 7.6C, the radial distribution function also show oscillating characteristics beyond 2.0Å, which most likely represent the liquid behavior of the system. Interestingly, a second distinct peak appears at 2.5 Å in the N-O *g(r)* plot when the temperature is at 4,000 K. This is due to the formation of HNCO molecules (isocyanic acid), a known detonation product. At higher temperatures, however, it's unlikely for this product molecule to exist, as entropic effects favor smaller molecules.

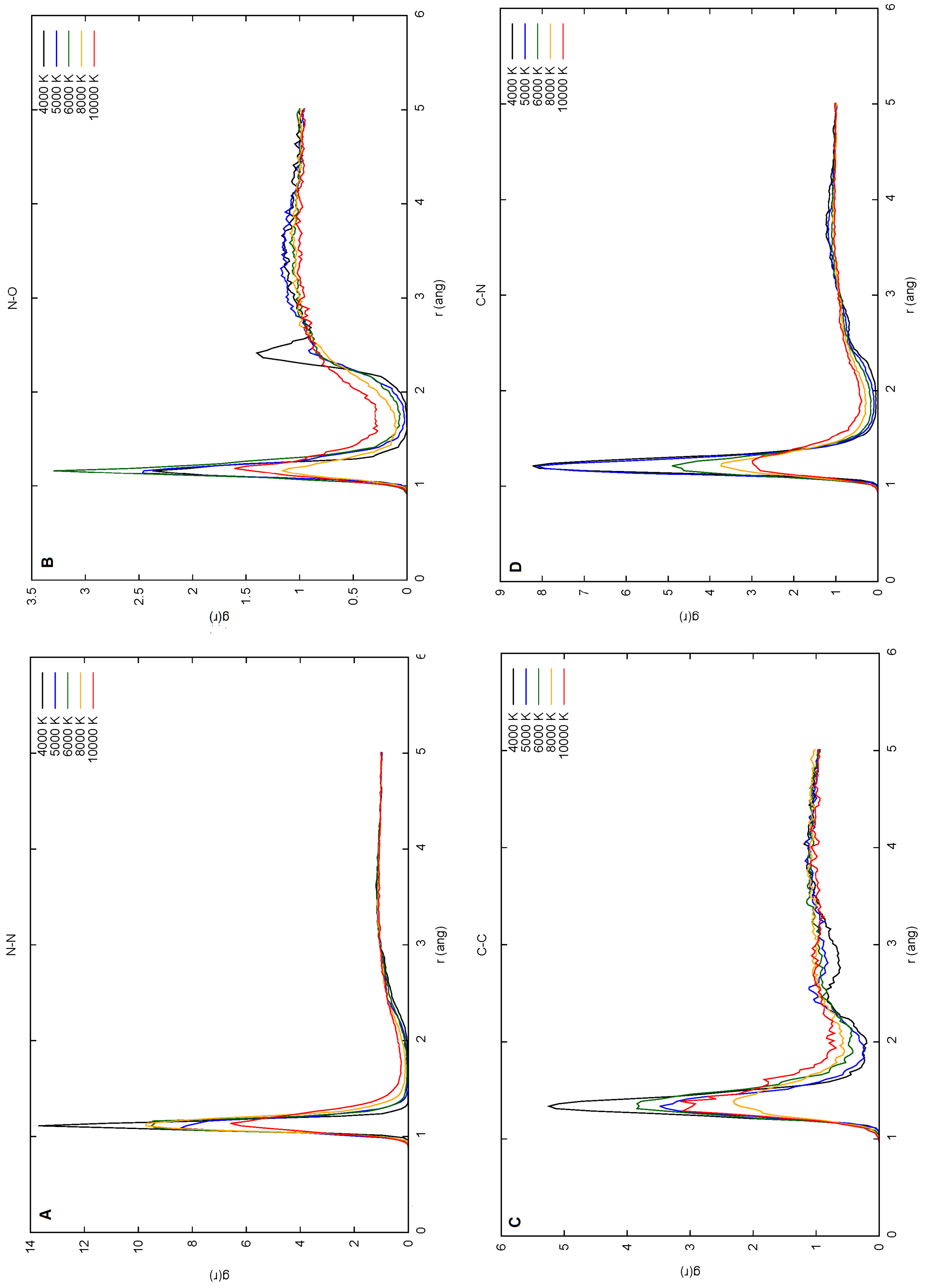


Figure 7.6: Radial distribution function analysis $g(r)$ on the MD simulation data from a system of 6 DAAF molecules at various temperatures at 1.0 g/cc. Each plot shows $g(r)$ for pairs of particles of two given types: A) nitrogen-nitrogen, B) nitrogen-oxygen, C) carbon-carbon, D) carbon-nitrogen.

7.3 Vapor Domes methods & results

Methods

To characterize the vapor-dome regime for a given material, we will first define a Maxwell construction in context and then define algorithms which use this construction to compute desired bounds for the phase-coexistence regime of the vapor dome.

Gibbs Free energy in Maxwell Construction We start with the thermodynamic identity for Gibbs Free Energy from Schroeder (1999)

$$dG = -SdT + VdP + \mu dN. \tag{7.2}$$

We will develop our algorithms to run on isotherms of fixed amounts of material, so $dT = 0$ and $dN = 0$. Thus this equation reduces to

$$dG = VdP \tag{7.3}$$

$$\left(\frac{\partial G}{\partial P}\right)_T = V \tag{7.4}$$

$$\tag{7.5}$$

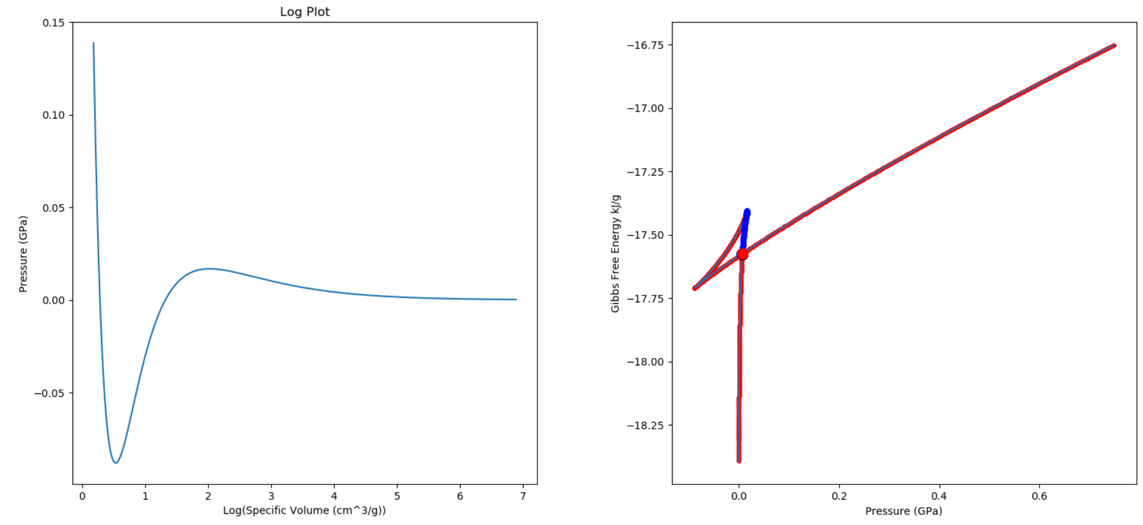


Figure 7.7: Left: Pressure versus $\log$(Volume) diagram produced by Magpie for H_2O at 550K. Right: Gibbs free energy versus Pressure for the same isotherm

We note by looking at the above diagram that there is a loop in Gibbs free energy for sub-critical isotherms. If we define the integral around this loop, we have

$$0 = \int_{\text{loop}} dG = \int_{\text{loop}} \left(\frac{\partial G}{\partial P}\right)_T dP = \int_{\text{loop}} VdP \tag{7.6}$$

$\int_{\text{loop}} V dP$ can be expressed as an integral in Volume versus Pressure – the curve enclosed by the modeled V-P diagram and a vertical line through the pressure P_g at which the loop intersects. We can transpose this diagram and then by picking a P_g, we can derive horizontal bounds V_0 and V_1 for the first and last intersection of the horizontal line at pressure P_g across the P-V diagram, thus yielding the following characterization determined by P_g

$$0 = \int_{V_0}^{V_1} P(V) - P_g dV \tag{7.7}$$

P_g satisfying this equation will necessarily be the vapor pressure for the coexistence regime of the material at this temperature.

Determining an appropriate volume grid for computations An evenly spaced grid in volume will lack resolution for sharp derivative changes at high pressures. For this reason, we will create a non-uniformly spaced grid based on segment lengths in the Gibbs free energy versus Pressure (G-P) Diagram.

```
function VOL_FILL(n_vol, vol_range)
    seg_len = length of segments in $G - P$ diagram
    total_len = sum(seg_len)
    ppl = n_vol/(total_len)                ▷ Compute desired point density per unit length
    seg_num = seg_len.*ppl                 ▷ Compute number of points to add to each segment
    vol_range_out = zeros(sum(seg_num))
    for seg, i in enumerate(vol_range) do
        vol_range_out[seg.first_ind:seg.last_ind] = even grid of seg_num[i] points from seg.begin
to seg.end
    end forreturn vol_range_out
end function
```

Computation via Integration Bracketing We will use a bracketing approach to determine P_g satisfying equation (7.7) . Then we'll use that pressure to determine the corresponding volume bounds on the coexistence region.

```
function INT_BRACKET(iso, vol_range, n_iter = 100, p_g = 0, p_u = 25, p_b = 0, f_tol = 0.00001)
    prs = iso.get('P')
    prs_acc = pressures after derivative sign changes
    if len(prs_acc) $>$ 1 then p_b = prs_acc[0] p_u = prs_acc[1]
    end if
    if len(prs_acc $\leq$ 1 then return Failure    ▷ should only happen for super-critical materials
    end if
    for i in range(n_iter) do
        prs_bc = pressures before p_g crossings of P-V diagram
        if len(prs_bc) == 2 — (len(prs_bc) == 0 then
            p_b = p_g
            p_g = (p_g + p_u)/2
        else if len(prs_bc) == 3 then
            pi = integral from first intersection
            if abs(pi) ¡ f_tol then return p_g and first and last intersection volumes
            else if pi $<$ 0 then
                p_u = p_g
                p_g = (p_g + p_b)/2
            else if pi $>$ 0 then
                p_b = p_g
                p_g = (p_g + p_u)/2
            end if
        else if len(prs_bc) == 1 then
            if p_g ¡ prs_acc[0] then
                p_b = p_g
                p_g = (p_g + p_u)/2
            else
                p_u = p_g
                p_g = (p_g + p_b)/2
            end if
        end if
    end forreturn Failure    ▷ We made it all the way here and still the intergral didn't converge
end function
```

Computation via Gibbs Free Energy intersection As we noted above, the self-intersection of the material plot in Gibbs free energy versus pressure (G-P) corresponds with the Maxwell bounds we seek. We will compute the Maxwell bounds by using a segment-intersection algorithm to compute the pressure at which the G-P diagram intersects itself and then derive the bounding volumes from each leg of the intersection.

```
function GIBBS_COEX(iso, vol_range, n_iter = 100, p_g = 0, p_u = 25, p_b = 0, f_tol = 0.00001)
    prs = iso.get('P')
    gib = iso.get('G')
    ind_pr, ind_pl = list(), list()
    for i in range(len(prs) - 1) do
        vec = (prs[i - prs[i+1], gib[i]-gib[i+1]])
        vecs = vectors from point i to every point before
        vecs_u = normalized vecs
        angs = signed angles between vec and each vector in vecs
        if (ind_pr is nonempty) & (ind_pl is nonempty) then
            if we're on the left side of the segment from pt_pl to pt_pr then
                ind_c = i                    ▷ this means we have crossed the last segment
                pt_c = (prs[i], gib[i])
                pt_cc = (prs[i+1], gib[i+1])
            end if
        end if
        inds_ur = angles within [0, $\pi/2$]
        if len(inds_ur) ¡ 1 then
            Continue
            ind_ur = index of point with smallest angle on the front right
            pt_fr = point with smallest angle on the front right
        end if
        inds_ul = angles within [0, $\pi/2$]
        if len(inds_ul) ¡ 1 then
            Continue
            ind_ul = index of point with smallest angle on the front left
             pt_fr = point with smallest angle on the front left
        end if
        ind_pr, pt_pr = pt_fr, ind_ur
        ind_pl, pt_pl = pt_fl, ind_ul
    end for
    pt_it = seg_intersect((pt_pl, pt_pr), (pt_c, pt_cc)) return pt_it
end function
```

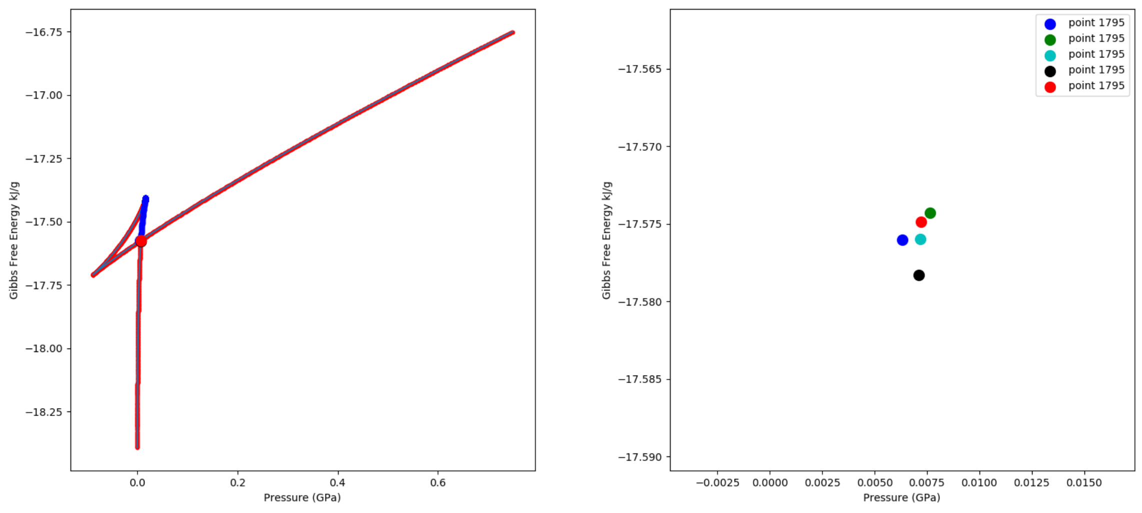


Figure 7.8: Left: Gibbs free energy versus Pressure for the same isotherm, note that the segment intersection search starts on the upper right and works around to the intersection point highlighted in red. Note that the blue segment highlights points that have a non-empty right and left side list of points – candidates for segment intersection. The red dot indicates the last such point – the intersection. Right: re-scaled view of the end-points of the segments at the identified intersection. Intersection is highlighted in red.

Computation of Critical Point We will compute the critical temperature by bracketing isotherm temperatures using conditions based on the pressure-volume diagram. We will define the critical temperature as the lowest temperature for which $\frac{\partial P}{\partial V}$ never changes sign (it should always be negative).

```
function TEMP_CRIT(t_g, i_mat, vol_range, n_iter=50, f_tol=0.00001, t_u=10000, t_b=0) ▷ start with guess t_g
    for i in range(n_iter) do
        iso = i_mat.isotherm(t_g, vol_range)
        prs = iso.get('P')                                    ▷ Compute derivative sign changes
        prs_dsd = abs(diff(sign(diff(prs))))
        prs_dbc = index(where prs_dsd ¿ 0)
        if len(prs_dbc) $>$ 0 then
            t_b = t_g
            t_g = (t_u + t_g)/2
        else if len(prs_dbc) $<$ 1 then
            t_u = t_g
            t_g = (t_b + t_g)/2
            if abs(t_g - t_gl) $<$ f_tol then return t_g
            end if
        end if
        if i == n_iter-1 then
            Print("Warning, max of iterations reached without convergence")
        end if
        t_gl = t_g
    end for
end function
```

Once we have computed critical temperature, we will compute the critical pressure and volume by searching for the spot where the $\left(\frac{\partial P}{\partial V}\right)_T$ is closest to 0. This will also be done by a bracketing procedure. We are essentially maximizing the derivative of pressure with respect to volume.

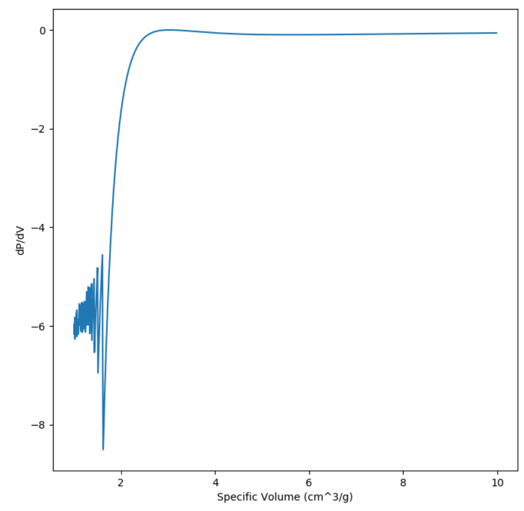


Figure 7.9: Approximated $\left(\frac{\partial P}{\partial V}\right)_{T_c}$ for the critical temperature at various volumes. The oscillations at the left side of the diagram are a numerical artifact of the derivative approximation for small volume intervals. Note that the diagram approaches 0 at its maximum – we will be computing the volume at which this maximum occurs via bracketing.

```
function PV_CRIT(mat, t_c, vol_range, n_iter=50, f_tol=0.00001)
    iso = mat.isotherm(t_c, vol_range)
    prs = iso.get('P')
    dp_dv = segment slopes in prs versus vol_range
    v_g = vol with largest dp/dv
    v_b = min(vol_range)
    v_u = max(vol_range)
    p_g = pressure with largets dp/dv
    p_b = min(prs)
    p_u = max(prs)
    dp_g = max(dp_dv)
    v_gl = first(vol_range)                                        ▷ keep last guess
    for i in range(n_iter) do
        v_l = (v_g + v_b)/2
        v_r = (v_g + v_u)/2                  ▷ try midpoints of right and left search windows
        vol_range = [v_l, v_r, v_g]
        iso = mat.isotherm(t_c, vol_range)
        prs_s = iso.get('P')
        p_g = prs_s[2]
        dp_g = ((p_g - p_b) + (p_u - p_g))/(v_u - v_b)             ▷ leapfrog approximation
        dp_l = ((p_g-prs_s[0])+(prs_s[0]-p_b))/(v_r-v_l)
        dp_r = ((p_u-prs_s[1])+(prs_s[1]-p_g))/(v_r-v_l)
        if dp_l > dp_g then                              ▷ If left side is better than guess
            v_u, p_u = v_g, p_g                               ▷ guess is new upper bound
        else                                                  ▷ if left is worse than guess
            v_b, p_b = v_l, prs_s[0]              ▷ Left upper bound is the new lower bound
        end if
        if dp_r > dp_g then                             ▷ If right side is better than guess
            v_b, p_b = v_g, p_g                               ▷ guess is new upper bound
        else                                                 ▷ if right is worse than guess
            v_u, p_u = v_r, prs_s[1]              ▷ Left upper bound is the new lower bound
        end if
        if abs(v_c - v_gl) < f_tol then
            break
        end if
        v_gl = v_g                                                      ▷ Save old guess
        v_g = (v_b + v_u)/2               ▷ set new guess in the middle of the updated bounds
    end for
     return dp_dv, p_g, v_g
end function
```

These algorithms converge with the following progression:

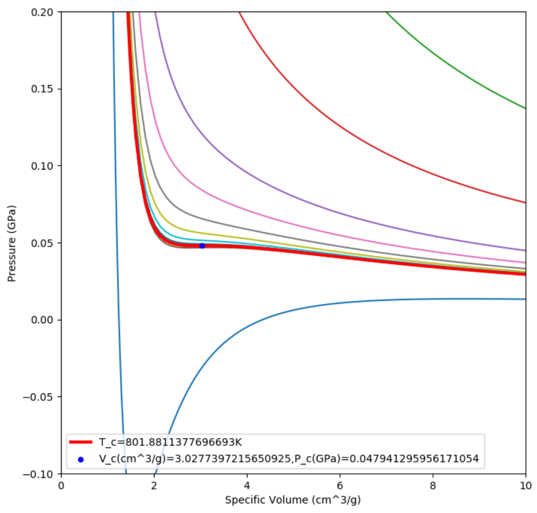


Figure 7.10: Final critical temperature isotherm is highlighted in Red, Critical pressure and volume are highlighted in Blue. The sequence of other isotherms are those processed during the bracketing procedure.

Results

Using the above algorithms, we computed bounds for coexistence regimes on a series of isotherms which were used to compose the following diagrams of the vapor dome for H_2O. These results are precise and can be computed about as quickly as Magpie can model isotherms at this grid size.

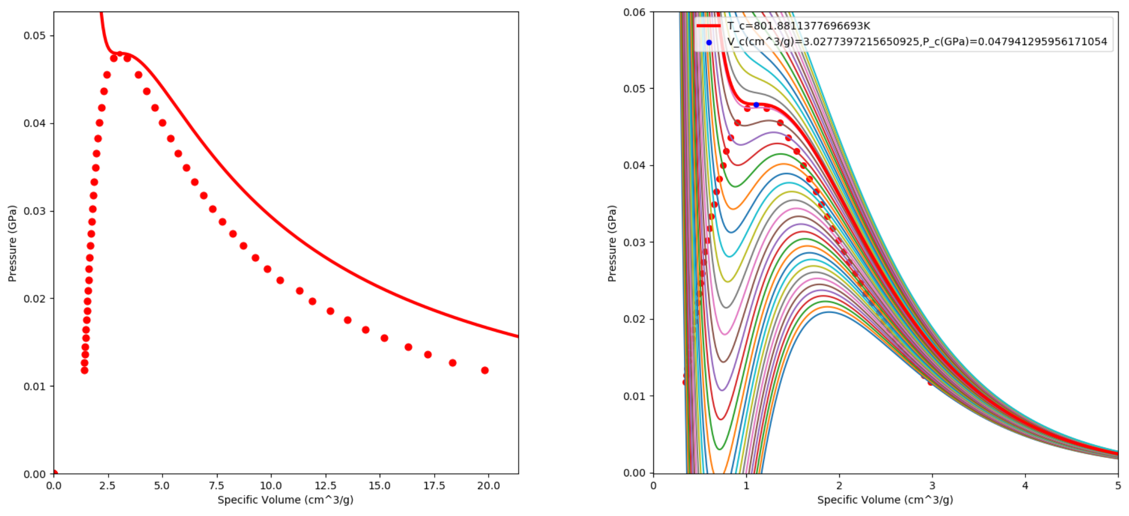


Figure 7.11: Left: Computed vapor dome for H_2O on Pressure versus Volume. Note that isotherm for critical temperature is highlighted in red. Right: Vapor Dome for H_2O on Pressure versus $\log$(Volume) with subcritical isotherms added. Note critical isotherm highlighted in red and critical point in blue.

Conclusion

We have developed generic algorithms for finding vapor domes for a given EOS, and these algorithms will be implemented into Magpie in the future. They will be used in an on-they-fly and post-processing fashion. The purpose of this work was to evaluate a few different algorithms for finding vapor domes for efficient implementation in Magpie.

7.4 Conclusions

The work we presented here demonstrates the continuing effort to improve Magpie's description of reactive matter. We undertook two projects toward that goal. First, we used first principles techniques to benchmark high explosives EOS, and second we tested different numerical implementations of the Maxwell construction.

From AIMD simulations, we determined that the equilibrium pressure of 6 DAAF molecules at a density of 1.0 g/cc is consistent with Magpie's results at a temperature range of 4,000 K to 5,000 K. This is a good indication that the consistency between the two sets of output should hold at lower temperatures, such as between 2,000 K to 4,000 K, which is the most important regime for modeling behavior of HEs. Using radial distribution function analysis, we obtained some insight into what types of molecules are present at the equilibrium states. The next stage of this study is to perform further AIMD simulation on more states so that statistics can be collected for a wide density-temperature space. It is also desirable to carry out AIRxMC for comparisons of the two simulations' results with each other and with Magpie.

For multiphase solver, We have written algorithms which achieve characterization of vapor domes for H_2O at a range of sub-critical temperatures. The algorithms do not depend on the material in question, although some adjustment may be necessary to the bounds on the isotherm computation and search area to make sure the relevant features of the isotherm diagrams are present. This result is the first step on a path to resolving coexisting states while maintaining the benefits of the Ross perturbation models within magpie. Further work should test these models on other materials, refine them, and integrate them into Magpie.

***Nga Ying Lo** is a graduate of Stony Brook University where she earned a Bachelor of Science in Physics and Astronomy. As an undergrad, she estimated the halo mass of approximately 1000 ultra diffuse galaxies to show that these abnormally large and dim galaxies are dark matter dominated. This fall, she'll continue her study in computational physics at the University of Copenhagen in Denmark.*

***Brian Bell** is a fourth year graduate student at the University of Arizona researching the theoretical foundations of machine-learning. Brian worked in industry for 4 years before going to graduate school, writing price optimization and demand forecasting software for clients including Homedepot and Sprouts. His timed markdown pricing algorithm controlled prices on approximately 2 billion dollars of retail inventory.*

Bibliography

M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, 1987. . See section 6.4.1.

P. E. Blöchl. *Phys. Rev. B*, 50:17953, 1994.

W. G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Phys. Rev. A*, 31(3): 1695–1697, 1985.

E. C. Koch. *Propellants, Explosives, and Pyrotechnics*, 41:526, 2016.

G. Kresse and J. Furthmüller. *Phys. Rev. B*, 54:11169, 1996.

G. Kresse and J. Furthmüller. *Comput. Mater. Sci.*, 6:15, 1996.

G. Kresse and J. Hafner. *Phys. Rev. B*, 47:558, 1993.

G. Kresse and J. Hafner. *Phys. Rev. B*, 49:14251, 1994.

G. Kresse and D. Joubert. *Phys. Rev. B*, 59:1758, 1999.

N. D. Mermin. *Phys. Rev.*, 137:A1441, 1965.

S. Nosé. A unified formulation of the constant temperature molecular-dynamics method. *J. Chem. Phys.*, 81(1):511–519, 1984.

J. P. Perdew, K. Burke, and M. Ernzerhof. Generalized gradient approximation made simple. *Phys. Rev. Lett.*, 77:3865, 1996. An erratum was published as Phys. Rev. Lett. 78, 1396 (1997).

Marvin Ross. A high-density fluid-perturbation theory based on an inverse 12th-power hard-sphere reference system. *J. Chem. Phys.*, 71(4):1567–1571, 1979. doi: 10.1063/1.438501. URL `https://doi.org/10.1063/1.438501`.

Daniel V Schroeder. An introduction to thermal physics, 1999.

Christopher Ticknor, Stephen A. Andrews, and Jeffery A. Leiding. Magpie: a new thermochemical code. *AIP Conf. Proc.*, (In Press), 2020.

Chapter 8

Direct Numerical Simulation of the Taylor-Green Vortex

Team Members
Joey Taylor

Mentors
Daniel Israel

Abstract

The direct numerical simulation (DNS) of the Taylor-Green vortex is investigated for a range of initial Reynolds number flows using a 256^3 grid. Calculations of the energy dissipation ε and skewness factor S agree with previous work by Brachet et al. (1983). The turbulent Reynolds number $R_t \approx$ Re after turbulent transition. $G/\sqrt{R_t}$ approaches a constant independent of Re for $t > 10$. Using S and G in k-ε modeling can reproduce DNS k and ε through turbulent transition. A model for both S and the coefficient of enstrophy destruction G is required to reproduce DNS results by solving the k-ε ODEs.

8.1 Introduction

Laminar initial flows can transition to turbulent flows via vortex-line stretching and folding, which generates vortices at smaller and smaller scales (Brachet et al., 1983). Predicting the behavior of statistical properties through this transition is difficult and not well-understood. One approach to improving models of transition requires the use of direct numerical simulation (DNS) to directly solve the Navier-Stokes (N-S) equation and calculate statistical quantities of interest. In this work, we implement a new Python code package (Mortensen and Langtangen, 2016) to simulate solutions to N-S, and examine statistical quantities of turbulent flows for different Reynolds (Re) numbers. We then use these statistical measures of the flow to evaluate a new version of a k-ε model. The k-ε model used is better able to capture the peak in energy dissipation during turbulent transition, and the DNS data generated should guide development of model parameters.

8.2 Methods

8.2.1 The Taylor-Green Vortex

One simple system that showcases the transition from laminar to turbulent flow is the Taylor-Green vortex (TGV). This flow is initialized as a single mode vortex, which then generates increased vorticity and small-scale eddies via vortex stretching and folding. The version of the initial conditions used in this work are stated below (Taylor and Green, 1937):

$$v_x = \sin(x)\cos(y)\cos(z) \tag{8.1}$$
$$v_y = -\cos(x)\sin(y)\cos(z) \tag{8.2}$$
$$v_z = 0 \tag{8.3}$$

As shown, this initial flow will undergo a turbulent transition before reaching a turbulent equilibrium roughly for $t > 10$.

8.2.2 SpectralDNS

The code package used in this work is spectralDNS, a Python package for simulating solutions to N-S. This publicly available code is written entirely in Python, while using mpi4py routines to perform the FFTs and multi-core communication in C. The resulting solver is less than 100 lines of Python code while retaining near-C++ level computational performance. This code is developed and maintained by Mortensen and Langtangen (2016).

8.2.3 k-ε Transition Modeling

One approach to understanding turbulent transition is to study the evolution of the kinetic energy (k) and the energy dissipation (ε) of the fluid system.

A statistical description of decaying isotropic turbulence is stated by the energy and dissipation equations (Ristorcelli, 2003):

$$\dot{k} = -\varepsilon \tag{8.4}$$
$$\dot{\varepsilon} = -\frac{\varepsilon^2}{k}\left(\frac{7}{3\sqrt{15}}SR_t^{1/2} + \frac{7}{15}G\right) \tag{8.5}$$

Which are originally stated in another form by Karman and Howarth (1938). Here, the turbulent Reynolds number R_t is calculated as

$$R_t = \frac{k^2}{\nu\varepsilon} \tag{8.6}$$

and the equations for k and ε can be numerically solved given a choice for S and G. Ristorcelli (2003) analyzed these equations for a constant S and G, which is predicted theoretically for isotropic high-Re turbulent flows.

We examine the insertion of S and G as calculated via DNS into 8.5, where S and G are evaluated as

$$S = \frac{6\sqrt{15}}{7}(\frac{\nu}{\varepsilon})^{3/2}\overline{\frac{\partial u_i}{\partial x_k}\frac{\partial u_i}{\partial x_l}\frac{\partial u_l}{\partial x_k}} \tag{8.7}$$

and

$$G = \frac{15}{7}\frac{\nu^2 k}{\varepsilon}\overline{\frac{\partial}{\partial x_l}\left[\frac{\partial u_i}{\partial x_k}\right]\frac{\partial}{\partial x_l}\left[\frac{\partial u_i}{\partial x_k}\right]} \tag{8.8}$$

respectively. While the isotropic values of S and G are roughly constant, during the transition the flow is highly anisotropic and, as such, S and G vary significantly.

8.3 Results

8.3.1 Verification of Previous Data

As a verification of spectralDNS and our implementation of the TGV, we reproduced data for ε and S across a range of Re from 100 to 10,000. While the 10,000 and likely the 5,000 Re cases are under-resolved with a 256^3 resolution, the other cases match closely with the Brachet et al. (1983) results.

The DNS dissipation ε is plotted in figure 8.1. ε is calculated as

$$\varepsilon = \frac{1}{2}|\nabla \times \vec{u}|^2. \tag{8.9}$$

Compared with figure 7 from Brachet et al. (1983), this simulation captures that correct dissipation peak heights and locations, while matching the overall curve for all Re up to 3,000. Furthermore, our results are extended out to $t = 20$, showing a decrease in ε across all Re for $t > 10$.

In figure 8.2, we show the evolution of S across the same set of Re. This metric also agrees with Brachet et al. (1983), indicating that our DNS is giving the same flow evolution as Brachet's. For $t > 7.5$, $S \approx 0.5$, which matches with the expected turbulent equilibrium value of S.

8.3.2 Calculations of G and the Turbulent Reynolds Number R_t

G is shown in figure 8.3. While it was expected that G is roughly constant at late time, the lack of a universal constant was initially puzzling. Additionally, the increase in the peak height for increasing Re was not expected.

An explanation for this likely comes from Speziale and Bernard (1992), where in equation (47) it is established that while G is a constant at late t, it will also scale with the square root of the turbulent Reynolds number R_t. We plot R_t in figure 8.4. While R_t decreases significantly during transition, the late t values level off to a constant for high Re. We further show the ratio of R_t and Re approaches order unity at late time across all initial Re in figure 8.5.

In figure 8.6 we plot the ratio $GR_t^{-1/2}$. From Speziale and Bernard (1992), the expected turbulent equilibrium value of $GR_t^{-1/2}$ is a constant independent of Re, which we observe computationally. For high Re, $GR_t^{-1/2}$ is approximately equal to 0.45 for $t > 10$. Further study into the match between this constant and the value predicted by Speziale and Bernard (1992) is required.

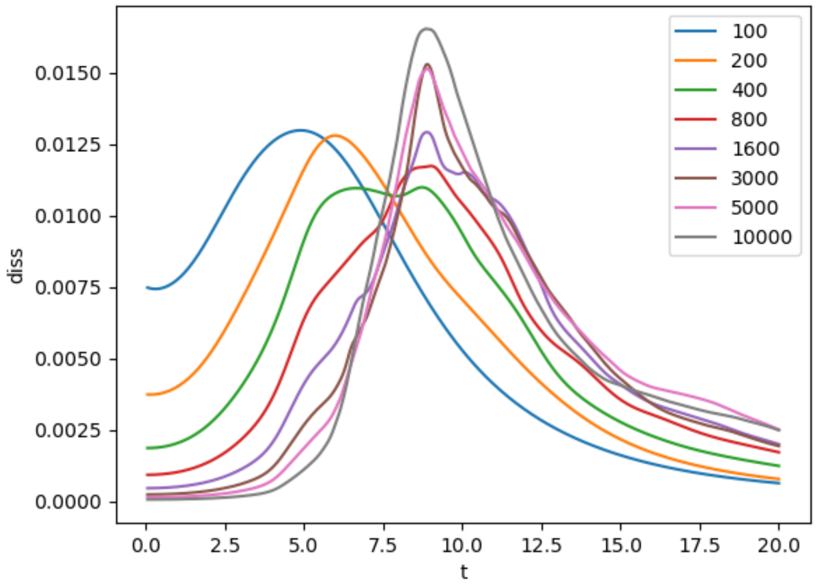


Figure 8.1: The dissipation ε plotted as a function of time. Each curve represents a different initial Reynolds number, which is listed in the legend at the top-right. Dissipation peaks at non-zero time during the turbulent transition. Compare with figure 7 of Brachet et al. (1983)

8.3.3 Evaluation of k-ε System with DNS Information

The evolution of k and ε in 8.5 depends on the model used for S and G. In figure 8.7, the DNS calculations of S and G are input into the evolution of equation 8.5. The result is that k and ε closely follow their DNS calculated values, even through transition.

To address the effects of S and G individually, we also evolved 8.5 while holding S or G fixed as constants. In figure 8.8, our calculated values of G from DNS are input to 8.5, while S is treated as a constant $S = 1/2$. This value was chosen as it is the late t turbulent equilibrium value for S observed in figure 8.2 and analyzed in the literature (Ristorcelli, 2003). The fit with DNS here is mixed, with some features fitting approximately while the curves themselves do not match well.

Figure 8.9 shows the evolution of ε using S from our DNS and a constant that approximates the late time value of G. Specifically, as $R_t \sim Re$ for $t > 10$, we approximated G by $\sqrt{Re}$ multiplied by an arbitrary constant C, where in figure 8.9 C = 0.8. This is aimed to recreate the late time constant of $GR_t^{-1/2}$, with some freedom to account for the lack of certainty around the constant value of both $GR_t^{-1/2}$ and R_t. Again, the fit here is mixed, with some features for some Re captured well, while the fits overall do not match with the DNS data.

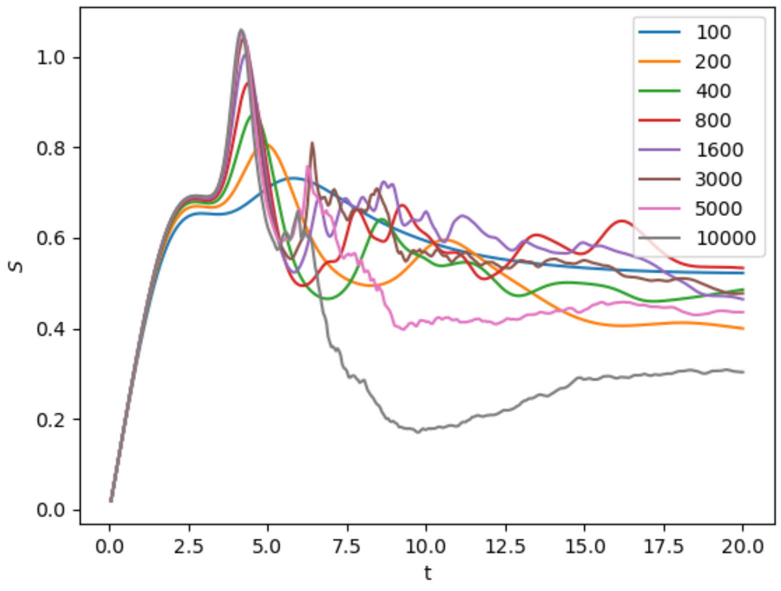


Figure 8.2: The velocity skewness S as a function of time. Each color curve corresponds to a different Re. S initially increases then plateaus around $t = 2.5$, before increasing to a peak during the turbulent transition. After this peak, each curve transitions to statistical noise around a mean value of roughly 0.5. Runs for Re = 5,000 and 10,000 are under-resolved, which likely explains why the curves dip much lower than other Re.

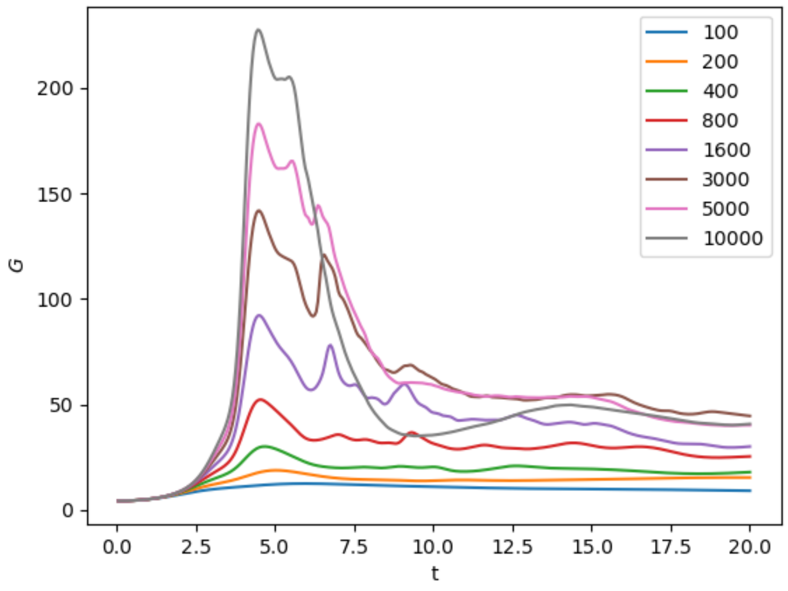


Figure 8.3: The coefficient of enstrophy destruction G plotted as a function of time. The Re used in each run is shown in the legend. Note that both the peak and late-time constant behavior of G increases as Re increases.

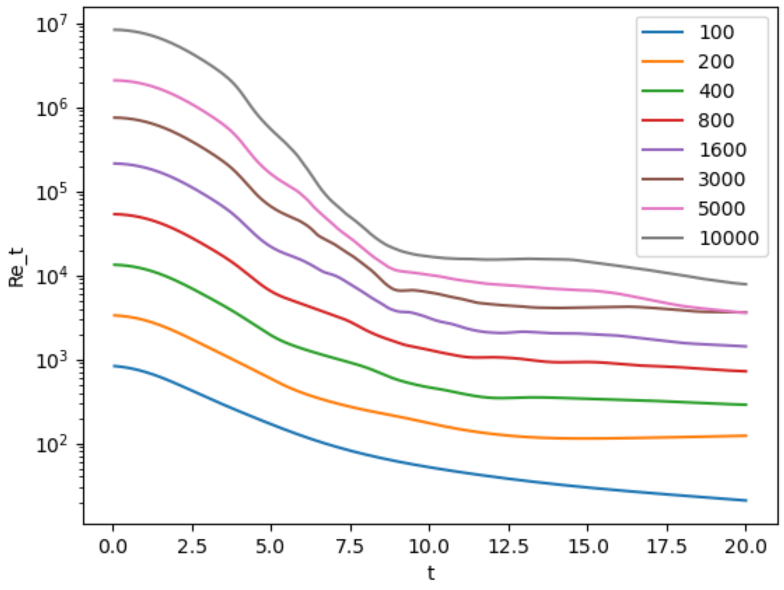


Figure 8.4: The turbulent Reynolds number plotted on a log scale. Each color curve refers to the initial Re of the run. For $t > 10$, $R_t \approx$ constant for $\mathrm{Re} > 100$.

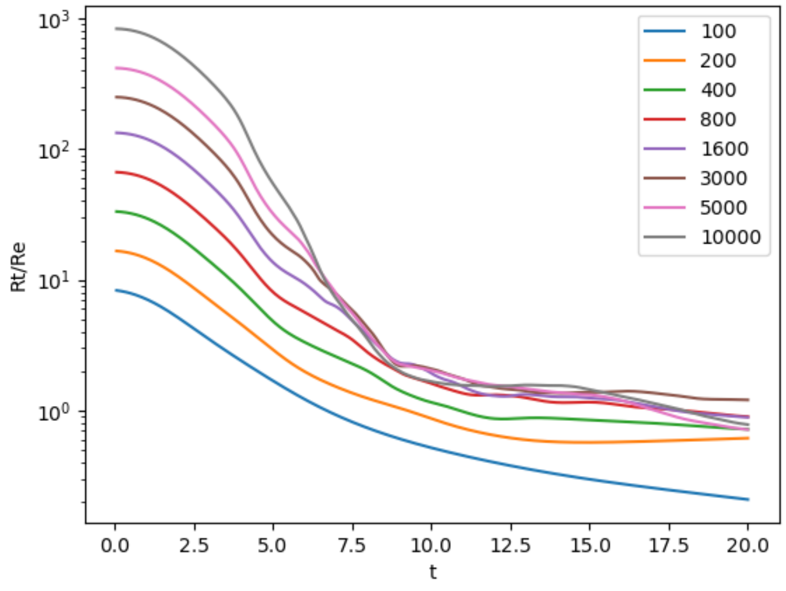


Figure 8.5: Turbulent Reynolds number plotted as the ratio R_t/Re, where Re is the initial Reynolds number. Note the late time behavior for $Re > 200$, as R_t/Re approaches unity.

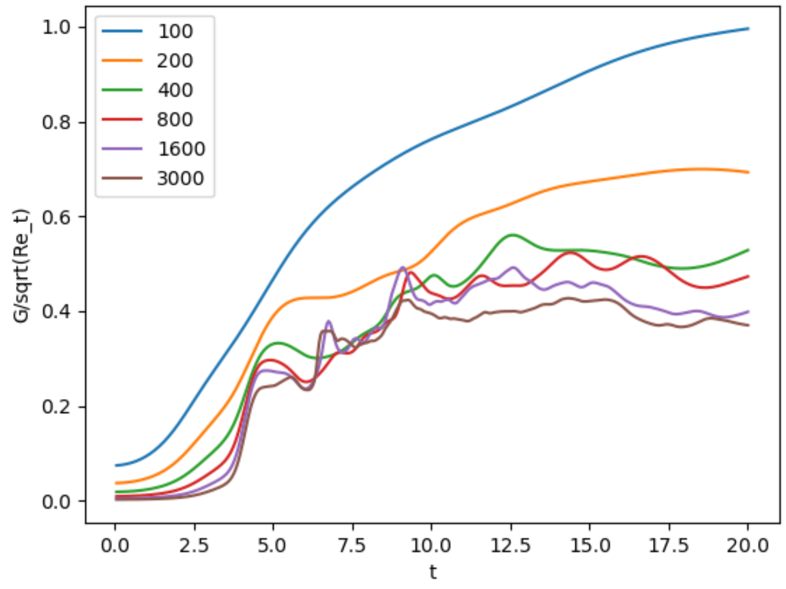


Figure 8.6: $GR_t^{-1/2}$ is plotted for a range of initial Re. For $\mathrm{Re} > 200$, note the close agreement for different flows through the turbulent transition. Additionally, the different runs appear to reach a roughly constant value for time $t > 10$. This indicates that $GR_t^{-1/2}$ may be a universal constant for flows in turbulent equilibrium.

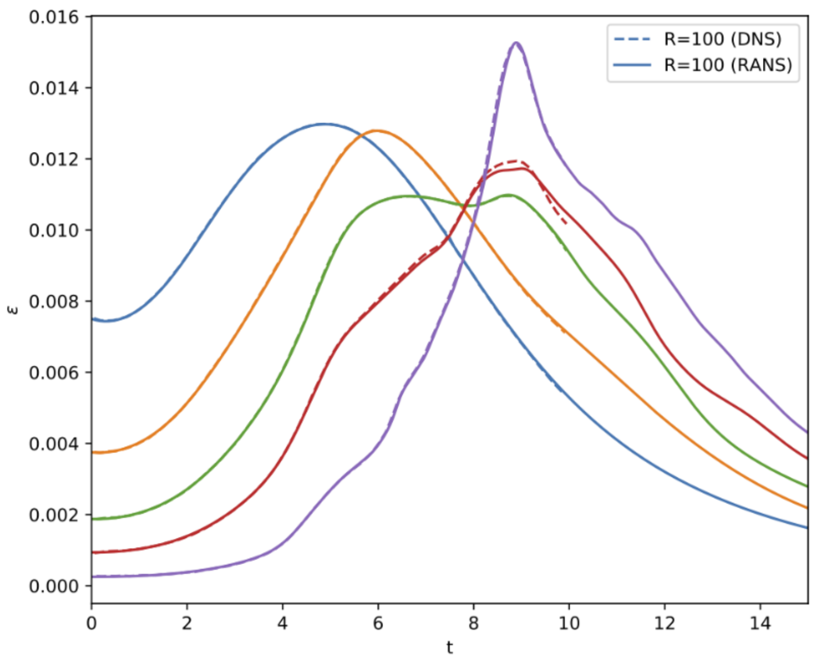


Figure 8.7: The solution to 8.5 for ε using S and G as calculated using DNS is plotted for different Re. The colors of Re used are identical to previous plots. The solid curve represents the solution to the ODEs, while the dashed curve comes from our DNS data of dissipation.

8.4 Conclusions

SpectralDNS provides a code framework for efficiently solving general Navier-Stokes systems using Python. This code is easy to use, maintain, and modify, and as such deserves consideration for DNS projects moving forward.

S approaches the predicted turbulent equilibrium value for large t. While shown by Brachet et al. (1983), our calculations further support that S approaches 1/2 for $t > 10$ in the TGV.

$GR_t^{-1/2}$ approaches a constant at late t for large Re, where the constant is independent of Re. This constant is roughly equal to 0.4 for the TGV. Additionally, during transition the high Re cases match remarkably well. The behavior of $GR_t^{-1/2}$ both during transition and during turbulent equilibrium warrants further study, particularly for $k - \varepsilon$ modeling.

R_t approaches Re at late t. Above $\mathrm{Re} = 200$ and for $t > 10$, R_t/Re became roughly constant with time and approached unity. This provides a useful limiting behavior for the turbulent Reynolds number in turbulent equilibrium in the TGV.

Running $k - \varepsilon$ with DNS values for S and G produces k and ε curves that agree with DNS results. When using well-informed late time constants for either S or G in place of DNS data, results are significantly stronger than using only the constants. While still not approaching all-DNS quality matches, this does indicate that further investigation of these constants is necessary. From these results, we conclude that a model for both S and G is necessary to fully model transition using this $k - \varepsilon$ model.

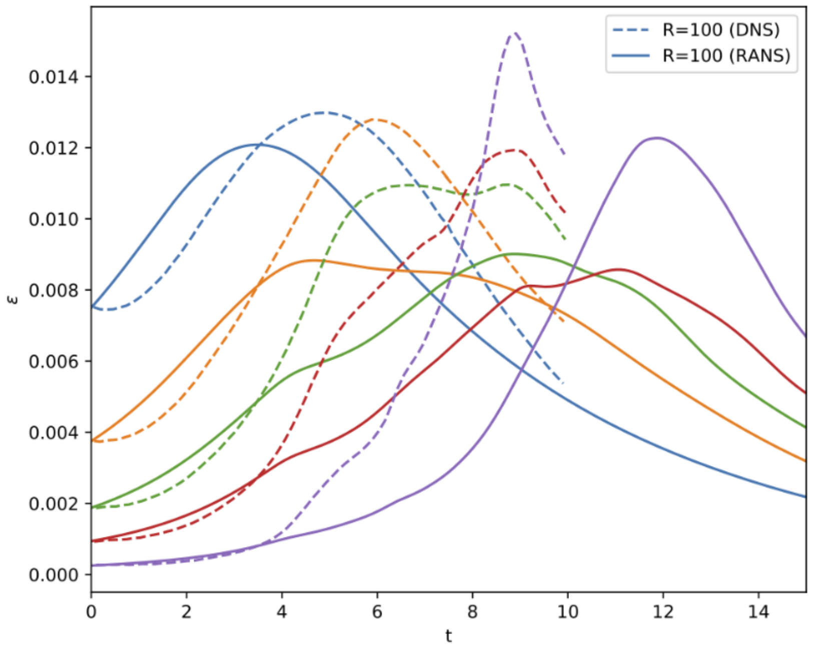


Figure 8.8: The solution to 8.5 for ε using G as calculated using DNS and a constant value of 1/2 for S. The colors of Re used are identical to previous plots. The solid curve represents the solution to the ODEs, while the dashed curve comes from our DNS data of dissipation. Curves capture some of the interesting peak behavior during transition, but overall are a poor fit to the DNS data.

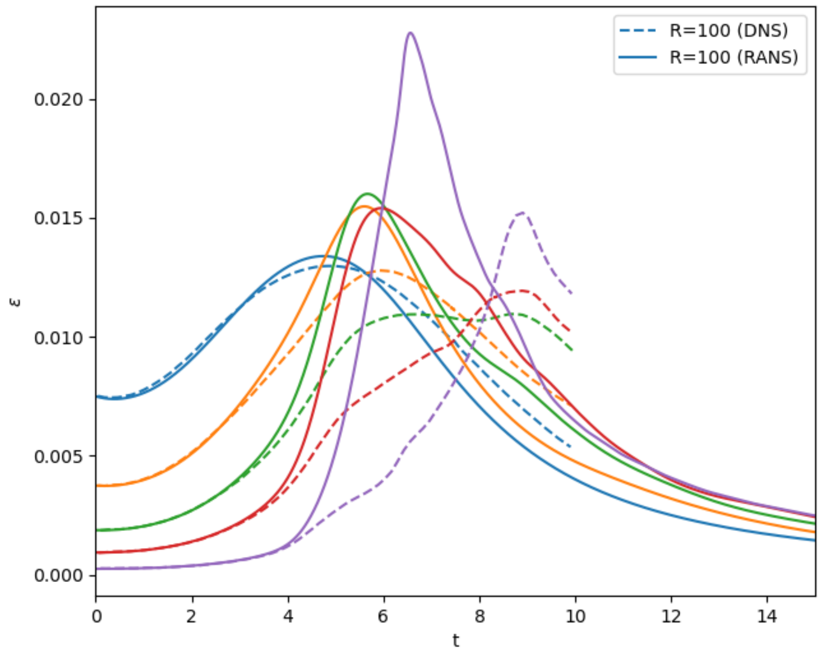


Figure 8.9: The solution to 8.5 for ε using a constant value of $0.8 * \sqrt{Re}$ for G and S as calculated using DNS. The colors of Re used are identical to previous plots. The solid curve represents the solution to the ODEs, while the dashed curve comes from our DNS data of dissipation. Similar to figure 8.8, the solution to 8.5 poorly fits with the data overall, though early t data fits well. This method also appears to work better for lower Re than for higher Re.

8.5 Acknowledgements

I would like to thank Daniel Israel for working with me this summer. I would also like to thank Daniel, Garry, Madison, and the other LANL staff who made this workshop possible. Thank you for this great (remote) research experience.

***Joey Taylor** graduated with a B.S. in physics and astronomy from University of Maryland in May 2020. He researched efficient computational methods and the effects of tidal heating on habitable zone widths as an undergraduate. Following a gap year away from class traveling and working on trail maintenance, Joey will begin work towards a PhD in atmospheric science in Fall 2021.*

Bibliography

M Brachet, D Meiron, and S Orszag. Small-scale structure of the taylor-green vortex. *Journal of Fluid Dynamics*, 130:411–452, 1983.

T Karman and L Howarth. On the statistical theory of isotropic turbulence. *Proceedings of the Royal Society of London A*, 164:192, 1938.

M Mortensen and H Langtangen. High performance python for direct numerical simulations of turbulent flows. *Computer Physics Communications*, 203:53, 2016.

J Ristorcelli. The self-preserving decay of isotropic turbulence: Analytic solutions for energy and dissipation. *Physics of Fluids 15, 3248*, 15:3248, 2003.

C Speziale and P Bernard. The energy decay in self-preserving isotropic turbulence revisited. *Journal of Fluid Mechanics 241, 645*, 1992.

G Taylor and A Green. Mechanism of the production of small eddies from large ones. *Proceedings of the Royal Society of London*, 158:499, 1937.

Chapter 9

Code Verification for MCNP Unstructured Mesh Geometry

Team Members
Harrison Leiendecker and Alex Warhover

Mentors
Jerawan Armstrong and Jim Ferguson

Abstract

MCNP (Monte Carlo N-Particle) is a general-purpose, continuous-energy, generalized-geometry, time-dependent, particle transport code. In order to run MCNP simulations, nuclear systems of interest must be modeled using constructive solid geometries (CSG) and/or unstructured mesh (UM) geometries. The UM model is a collection of finite elements and nodes generated by meshing tools, such as Abaqus or Cubit. The CSG and UM particle tracking algorithms are implemented differently in the MCNP code. Unlike the CSG feature, the MCNP UM feature is not well verified. This project proposes to study some selected equivalent CSG and UM models. For example, the benchmarks of the Godiva critical assembly of HEU and Oktavian 14 MeV neutron source will be modeled and analyzed. The simulation results from these two different models will be compared to verify the MCNP UM code implementations. Students will learn how to model and simulate the selected nuclear systems in MCNP and analyze the simulation results.

9.1 Introduction

Los Alamos National Laboratory's (LANL) Monte Carlo N-Particle (MCNP) [1] is a general-purpose, continuous-energy, generalized-geometry, time-dependent, particle transport code (Werner et al.,

[1]MCNP® and Monte Carlo N-Particle® are registered trademarks owned by Triad National Security, LLC, manager and operator of Los Alamos National Laboratory for the U.S. Department of Energy under contract number 89233218CNA000001. Any third party use of such registered marks should be properly attributed to Triad National Security, LLC, including the use of the ®designation as appropriate. Any questions regarding licensing, proper use, and/or proper attribution of Triad National Security, LLC marks should be directed to trademark@lanl.gov. For the

2017). The collection of MCNP references and user manuals can be found at `https://mcnp.lanl.gov`. Complex nuclear systems that cannot be modeled by computer codes that implement deterministic methods are typically studied via Monte Carlo simulations. In order to perform MCNP simulations, geometries representing nuclear systems of interest must be defined in MCNP input files. The MCNP geometry is organized into cells bounded by surfaces, and this geometry is known as a constructive solid geometry (CSG). It is very difficult and time-consuming to build CSG models for complicated geometry problems.

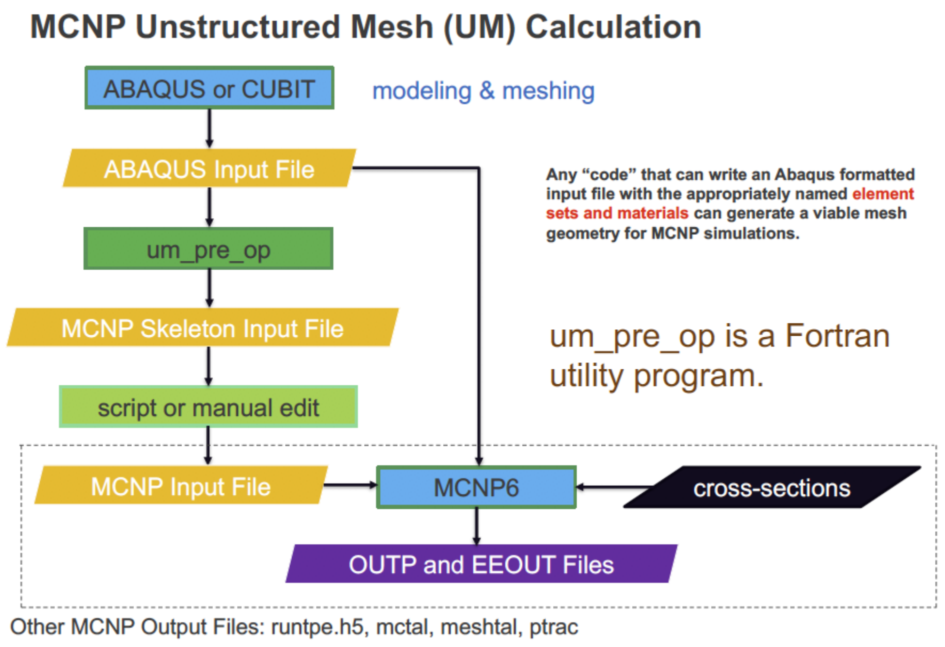

MCNP Unstructured Mesh (UM) Calculation
ABAQUS or CUBIT
modeling & meshing
ABAQUS Input File
Any "code" that can write an Abaqus formatted input file with the appropriately named element sets and materials can generate a viable mesh geometry for MCNP simulations.
um_pre_op
MCNP Skeleton Input File
um_pre_op is a Fortran utility program.
script or manual edit
MCNP Input File
MCNP6
cross-sections
OUTP and EEOUT Files
Other MCNP Output Files: runtpe.h5, mctal, meshtal, ptrac

Figure 9.1: **MCNP Unstructured Mesh Calculation Using UM_PRE_OP Program**

MCNP version 6 (also known as MCNP6) has the capability for tracking particles on unstructured mesh (UM) geometry models (Martz and Crane, 2012; Martz, 2012). The MCNP UM feature has been developed for performing calculations of complex geometry models. This capability is for tracking particles on the hybrid geometries where the finite element meshes are embedded into CSG cells. The UM geometry model is the collection of finite elements and nodes generated by mesh generation software packages, such as Abaqus/CAE software suite (`www.3ds.com/simulia`) or CUBIT Toolkit (`https://cubit.sandia.gov`). The MCNP UM feature was originally designed for UM models read from Abaqus input formatted files. The MCNP version 6.2.0 and later recognizes UM models read from Abaqus input files and MCNPUM formatted files converted from Abaqus input files. Other finite element analysis software packages may generate UM models and then convert these models into Abaqus input formatted files. A high-level of MCNP UM calculation is shown in Figure 9.1.

The MCNP code is the most used Monte Carlo particle transport code. Various verification and validation (V&V) suites have been developed and distributed with the MCNP code. The MCNP

purposes of visual clarity, the registered trademark symbol is assumed for all references to MCNP within the remainder of this report.

code has become the gold standard for Monte Carlo particle transport calculations since the users have the confidence in the code that the V&V suites provide. Despite the increasing use of MCNP UM calculations for new applications, there is no V&V suite of UM models distributed with the MCNP6 code. Therefore, the primary goal of this project is to convert the CSG models from MCNP V&V suite into UM models. With this goal, our main focus is to convert the Oktavian Benchmark in the CSG VERIFICATION_SHLD_SVDM suite into the UM models and verify the CSG and UM results.

In this report, first there will be a discussion of the files necessary to create an unstructured mesh in MCNP. These methods will be demonstrated on 2 simpler examples. The first is a simple cube, where we expect to get perfect agreement the different geometry representations. The second is the Godiva Benchmark, which we can use to observe how mesh quality effects the calculation result. We will then end with a detailed discussion of our work on the Oktavian Benchmark and the results thereof.

9.2 Monte Carlo Particle Transport Method

The MCNP code simulates individual particles to "solve" the Linear Boltzmann Transport Equations (LBTE). Monte Carlo method obtains the answers by simulating individual particles and recording some aspects (tallies) of their average behavior. The behavior of particles in the physical system is then inferred (using the central limit theorem) from the average behavior of the simulated particles. In order to simulate the nuclear system, several key assumptions are made: each neutron history is independent, all neutrons have the same probability density, geometry/material are fixed over time, collisions are instantaneous, particle speeds are small enough to ignore relativistic effects, and particle speeds are large enough to ignore quantum effects (Brown, 2016). For fixed source calculations, MCNP can effectively solve for the time-dependent LBTE. The time-dependent LBTE has seven variables: 3 position coordinates, 2 angles, energy, and time.

$$\begin{aligned}\frac{1}{v}\frac{\partial\psi(\vec{r},E,\vec{\Omega},t)}{\partial t} &= Q(\vec{r},E,\vec{\Omega},t)+\iint\psi(\vec{r},E,\vec{\Omega},t)\Sigma_s(\vec{r},E'\rightarrow E,\vec{\Omega}\cdot\vec{\Omega}')d\vec{\Omega}'dE' \\ &\quad +\frac{\chi(\vec{r},E)}{4\pi}\iint\nu\Sigma_F(\vec{r},E')\psi(\vec{r},E,\vec{\Omega},t)d\vec{\Omega}'dE' \\ &\quad -[\vec{\Omega}\cdot\nabla+\Sigma_T(\vec{r},E)]\cdot\psi(\vec{r},E,\vec{\Omega},t) \qquad (9.1)\end{aligned}$$

This full form of the time-dependent LBTE can be simplified and expressed as:

$$\frac{1}{v}\frac{\partial\Psi(\vec{r},E,\vec{\Omega},t)}{\partial t}=Q+[S+M]\Psi-[L+T]\Psi \qquad (9.2)$$

In this form of the equation, Q represents possible external sources. S represents scattering, M represents multiplication, and together they give the potential gains to the particle transport. L represents leakage, T represents collisions, and together they give the potential losses to the particle transport.

Alternatively, the time-independent LBTE can be reduced to an static eigenvalue problem.

$$[L+T]\Psi_{k_{eff}}(\vec{r},E,\vec{\Omega}) = [S + \frac{M}{k_{eff}}]\Psi_{k_{eff}} \tag{9.3}$$

In this steady state equation, k_{eff} is the effective multiplication factor. $k_{eff} = 1$ is a critical assembly, whereas $k_{eff} < 1$ is subcritical and $k_{eff} > 1$ is supercritical. In order to use the eigenvalue form of the equation, we assume that there is a fixed geometry and materials, there is no external source (Q = 0), and $\partial\Psi/\partial t = 0$. Using these assumptions, Equation 9.3 can be derived from Equation 9.2.

9.3 Abaqus Input File

An Abaqus input file is an ASCII file defining a finite element model. Three types of input lines are used in an Abaqus input file: comment lines, keyword lines, and data lines. Information on Abaqus input keywords and syntax rules can be found in Abaqus Analysis User's Guide. Note that Aabqus licensing is required in order to access Abaqus documentation. This section provides a brief description of the Abaqus keyword lines and data lines required to construct a MCNP UM model. See the example Abaqus input files in the Appendix 9.A-9.A for reference.

Comments in an Abaqus input file begin with stars in column 1 and 2 (**) and they are ignored by Abaqus and MCNP. The first non-blank character of each keyword line must be a star (*). Keyword lines may have parameters which appear as words or phrases separated by commas. The keyword must be followed by a comma (,) if it has parameters. Parameters in a keywords line can stand alone or have values. Most keyword lines require one or more data lines. If the data lines are required, they must immediately follow the keyword line. The data lines are used to provide numeric or alphanumeric entries for the associated keyword options.

An input file often begins with the *Heading option which is used to define a title for the analysis. After the heading the input file usually contains a model data. A model geometry is described by element and their nodes and can be defined by organizing into parts which are positioned relative to one another in an assembly. The MCNP code recognizes only an Abaqus model that is organized into an assembly of part instances. These geometry keywords and associated data lines are required for MCNP UM calculations: *Part, *Node, *Element, *Elset, *End Part, *Assembly, *Instance, *End Instance, and *End Assembly. In addition, MCNP makes use of the Abaqus definition of a set of elements to define a MCNP geometric cell. A line start with *Elset is used to group elements into an element set. The Elset option can be at a part level and an instance level, but only Elset a a part level is used by the MCNP code where the *Elset has to be a keyword line. Several restrictions are imposed on Elset names in models generated for MCNP UM calculations. See the MCNP user manual (Martz, 2012) for the syntax of Abaqus input file required by the MCNP code.

9.4 Pre-processing: write_mcnp_um_input.py

Figure 9.1 shows that two input files are necessary to run UM using MCNP: an Abaqus input file and a MCNP input file. The code um_pre_op is a Fortran utility program that was designed to process the mesh information from an Abaqus input file and write the corresponding MCNP input file for that mesh. Recently, the code write_mcnp_um_input.py was written as a Python replacement for the um_pre_op program. We compared the results from the output files from these two programs, where we found agreement between them. In addition, we found a bug in both programs that causes discrepancies in the densities of parts that are divided into many elsets. After comparing outputs for 40 Abaqus input files, we found that there was consistently a 12.56% difference between the densities in the parts with many elsets. The bug has since been fixed for write_mcnp_um_input.py. The bug has not been fixed for the um_pre_op program (writing MCNP input file option) as this function will be obsolete after write_mcnp_um_input.py is approved for public release.

9.5 Simple Cube

The simple cube model serves as another benchmark to compare the UM results to CSG results. The simple cube geometry was selected specifically because it has no curvature. For this reason, the mesh is able to perfectly replicate the geometry of the CSG. With this perfect agreement, we expect to get identical results when we compare UM to CSG calculations.

For these simulations, the cube was divided into 8 even regions, bisected along the three midplanes. With this geometry we were able to perform several variations to the experiment. For all iterations of the simple cube, we used fixed source calculations. In our first iteration, a single neutron source of 14 MeV was placed at the center of a cube of enriched uranium. A volume of 10^3 cm^3 and 10^6 source particles were used for all of the simple cube simulations. To measure the leakage from the cube, a tally was placed on a sphere surrounding the cube to count the surface current of neutrons that escaped the cube (Figure 9.2). As expected, we get perfect agreement between the CSG and UM models.

The simulation was then repeated for two variations. First, a neutron source (of 14 MeV) was placed at the center of each of the 8 cells that construct the cube instead of a single source at the center. The neutron surface current was again tallied on a sphere that surround the cube (Figure 9.3). Next the neutron source was instead placed on the surrounding sphere and directed inwards. In this case, we instead tallied the average neutron flux and energy deposition over the cells of the cube (Figures 9.4 and 9.5). In all of these cases, we once again found perfect agreement between the CSG and UM models.

These sets of simulations were then repeated for a neutron source of 1 MeV, a Maxwell fission spectrum, and a Watt fission spectrum. The simulations were also repeated with different materials in the cube: U-235 metal, Pu-239 metal, air, and void. In all of the iterations we performed, we continued to find perfect agreement between the CSG and UM models.

9.6 Godiva

The Godiva benchmark is a bare, 52.41 kg, homogeneous sphere of highly enriched uranium (HEU) with a radius of 8.7401 cm and enrichment of 93.71 wt.%. It is based on the Godiva experiment that was operated during the 1950s at Los Alamos Scientific Laboratory (LASL) (Paxton, 1983). The size of Godiva was determined by the critical size of previous uranium pseudospheres. The Godiva sphere is essentially the experimental definition of critical mass.

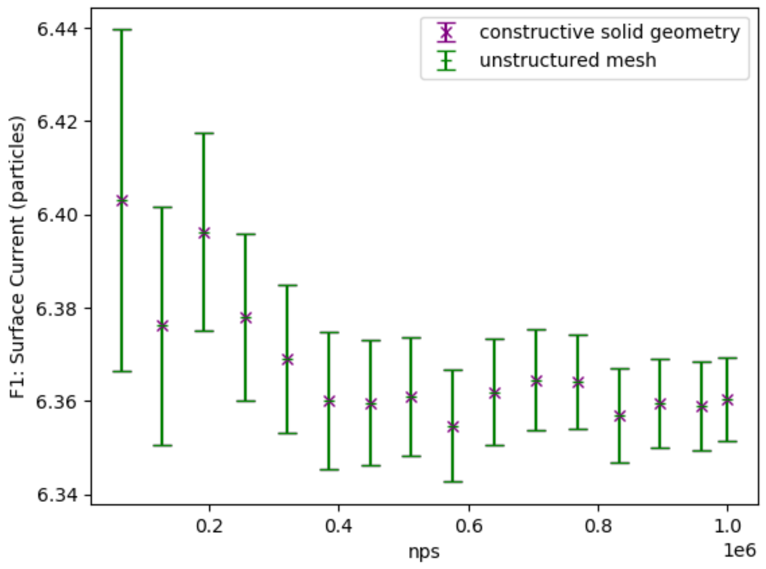


Figure 9.2: **Neutron surface current through a sphere around the simple cube with a single neutron source at the center of the cube**. The UM and CSG models have perfect agreement. Surface current is normalized to nps, so the current converges on a value as nps increases.

For this portion of the project, we compared the results from using an unstructured mesh of the Godiva sphere to a constructive solid geometry model. The Abaqus input files for the unstructured mesh were prepared in advance by Karen Kelly (E13). The UM version of the model was prepared with a variety of mesh types and elements numbers (Table 9.1). This variation of the mesh resulted in slightly different volumes among these UM models. The density of the model was fixed for all models, so there was a resulting change in mass between the models. Therefore, a CSG model of a sphere with the same volume as the UM was constructed for each case. In general, meshes with fewer elements tended to underestimate the mass. Furthermore, the type of mesh itself affected how closely the mesh resembled a sphere. The hexahedra element mesh uses elements that has six faces, as opposed to four faces in the tetrahedra element mesh. This allows a hexahedra mesh with the same number of elements as a tetrahedra mesh to more accurately replicate a sphere. The mesh could also be improved by using higher order elements. First order elements have nodes only at vertices. Second order elements have nodes at vertices and at midpoints along their edges. This allows a mesh with second order elements to more accurately replicate a sphere than a mesh with the same number of first order elements.

For our simulations, we find that meshes which more accurately replicate a sphere (via a larger number of elements, hexahedra meshing, and/or second order elements) had larger masses. As mass increases towards the mass of the original Godiva experiment, we found that k$_{eff}$ increases towards k$_{eff}$ $= 1$ (Figure 9.6). Using meshes that more closely resembled true spheres also gave closer agreement between the UM model and the CSG model of equivalent volume (Figure 9.7).

The unstructured mesh for the Godiva simulation allows us to calculate tallies for each mesh

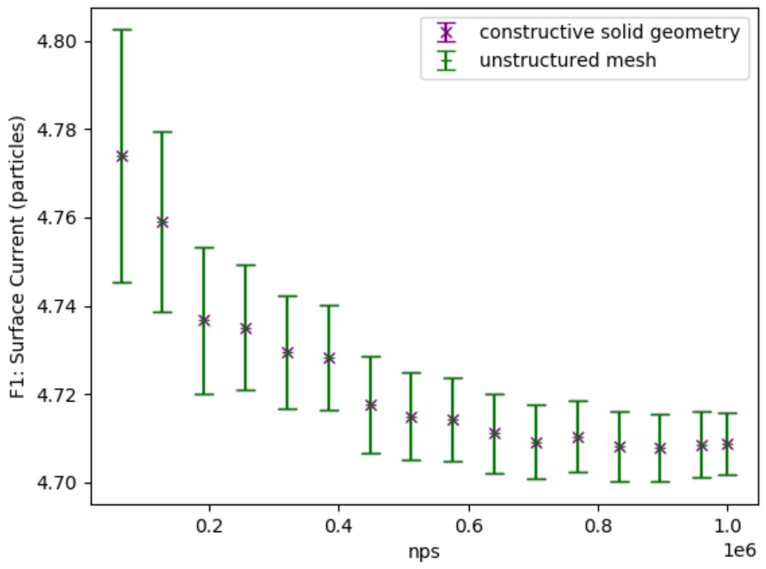


Figure 9.3: **Neutron surface current through a sphere around the simple cube with a neutron source at the center of each cell of the cube**. There are 8 cells in the cube, resulting in 8 neutron sources. The UM and CSG models have perfect agreement. Surface current is normalized to nps, so the current converges on a value as nps increases.

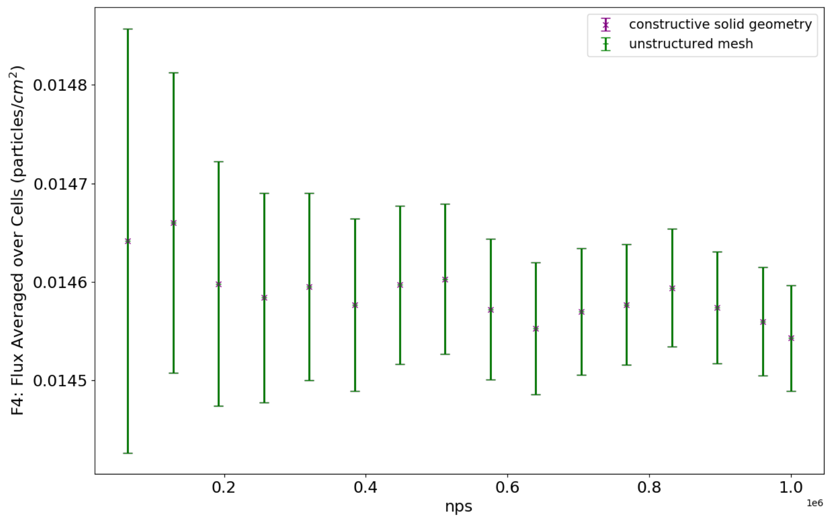


Figure 9.4: **Neutron flux in the simple cube from a spherical source surrounding the cube**. The UM and CSG models have perfect agreement. Flux is normalized to nps, so the flux converges on a value as nps increases.

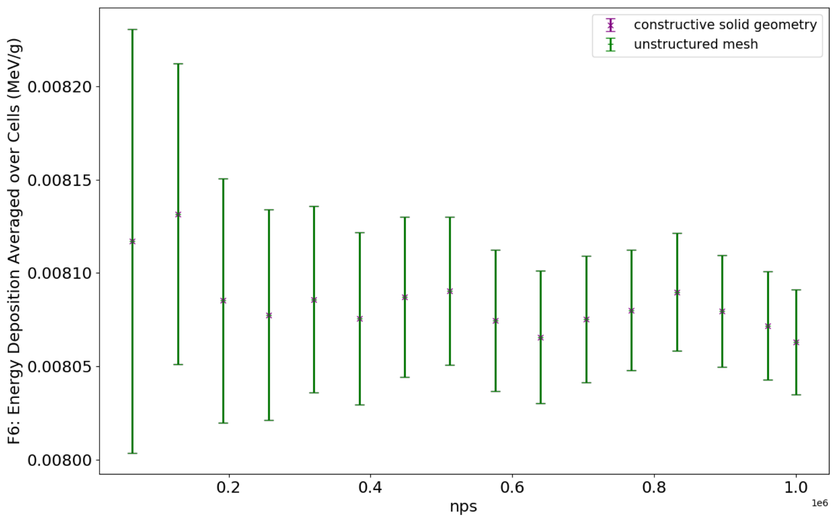


Figure 9.5: **Neutron energy deposition in the simple cube from a spherical source surrounding the cube**. The UM and CSG models have perfect agreement. Energy deposition is normalized to nps, so the value converges as nps increases.

element, instead of a single cell for the whole sphere. The script Convert_MCNP_eeout_to_VTK.py converts the MCNP UM output files to VTK files (Kulesza and McClanahan, 2019). The scientific visualization software ParaView (`https://www.paraview.org`) can then be used to visualize the MCNP UM results. Thus, we are able to get more detailed information on neutron flux as well as neutron and photon energy deposition throughout the entire sphere (Figures 9.8 and 9.9). Only minor variations are found between tallies of meshes with a sufficiently large enough number of meshes.

9.7 Oktavian Sphere

9.7.1 Background

The Shielding Integral Benchmark Archive Database (SINBAD) was produced in 1996. The benchmark is part of an ongoing project to provide experimental data and computational models that validate nuclear data libraries (Milocco, 2009). The MCNP code is the primary tool used to compile the computational database for SINBAD. The Oktavian experiments are a subset of the SINBAD database.

The Oktavian Benchmark measured the leakage current spectrum coming from the outer surface of a spherical pile of material. A 14 MeV D-T source was placed at the center of the pile. The experiment was performed using time-of-flight techniques to determine neutron energy (Milocco, 2009). The MCNP models were designed to best replicate the original specifications from the experiment while not including the source duct, collimator, or measurement apparatus of the

Element Type	Number of Elements	mass (g)	k$_{eff}$	rel. error
Tet1	32	36827.30	0.90038	0.00038
Tet1	193	44956.80	0.95656	0.00040
Tet1	459	48943.10	0.98121	0.00043
Tet1	4847	51556.80	0.99465	0.00041
Tet1	35453	52178.60	0.99883	0.00041
Hex1	32	40058.70	0.92381	0.00040
Hex1	56	43490.10	0.94678	0.00040
Hex1	160	48873.10	0.97995	0.00040
Hex1	896	51524.70	0.99489	0.00042
Hex1	7168	52198.50	0.99887	0.00044
Tet2	8	44607.30	0.95189	0.00040
Tet2	64	51642.10	0.99576	0.00046
Tet2	32	51642.10	0.99563	0.00044
Tet2	193	52275.30	0.99929	0.00041
Tet2	459	52391.70	0.99992	0.00038
Tet2	4847	52418.20	0.99903	0.00046
Tet2	35453	52419.80	0.99930	0.00042
Hex2	32	51642.10	0.99599	0.00041
Hex2	56	52058.60	0.99786	0.00042
Hex2	160	52365.30	0.99985	0.00046
Hex2	896	52416.80	0.99971	0.00040
Hex2	7168	52419.80	1.00009	0.00041

Table 9.1: **Mass, k_{eff}, and the relative error of k_{eff} from the UM models of the Godiva sphere**. The table is first sorted by the element and then by the mass of the model. Tet1 represents a first order tetrahedron, Tet2 represents a second order tetrahedron, Hex1 represents a first order hexahedron, and Hex2 represents a second order hexahedron.

UM Mass (g)	UM k$_{eff}$	UM rel. error	CSG Mass (g)	CSG k$_{eff}$	CSG rel. error
36827.30	0.90038	0.00038	36827.60	0.90492	0.00041
40058.70	0.92381	0.00040	40058.70	0.92745	0.00042
43490.10	0.94678	0.00040	43490.80	0.94919	0.00037
44607.30	0.95189	0.00040	44607.50	0.95615	0.00040
44956.80	0.95656	0.00040	44957.40	0.95785	0.00044
48873.10	0.97995	0.00040	48872.70	0.98048	0.00043
48943.10	0.98121	0.00043	48943.10	0.98080	0.00041
51524.70	0.99489	0.00042	51525.80	0.99466	0.00041
51556.80	0.99465	0.00041	51557.80	0.99393	0.00042
51642.10	0.99563	0.00041	51641.50	0.99529	0.00042
51642.10	0.99576	0.00044	51641.50	0.99529	0.00042
51642.10	0.99599	0.00046	51641.50	0.99529	0.00042
52058.60	0.99786	0.00042	52059.40	0.99863	0.00042
52178.60	0.99883	0.00041	52179.50	0.99909	0.00043
52198.50	0.99887	0.00044	52199.30	0.99940	0.00043
52275.30	0.99929	0.00041	52274.70	0.99907	0.00040
52365.30	0.99985	0.00046	52366.30	0.99882	0.00043
52391.70	0.99992	0.00038	52391.50	0.99912	0.00038
52416.80	0.99971	0.00040	52416.70	0.99935	0.00039
52418.20	0.99903	0.00046	52418.40	0.99964	0.00041
52419.80	0.99930	0.00041	52420.20	1.00084	0.00047
52419.80	1.00009	0.00042	52420.20	1.00084	0.00047

Table 9.2: **A comparison of mass, k_{eff}, and the relative error of k_{eff} between the UM and CSG cases for the Godiva sphere.**

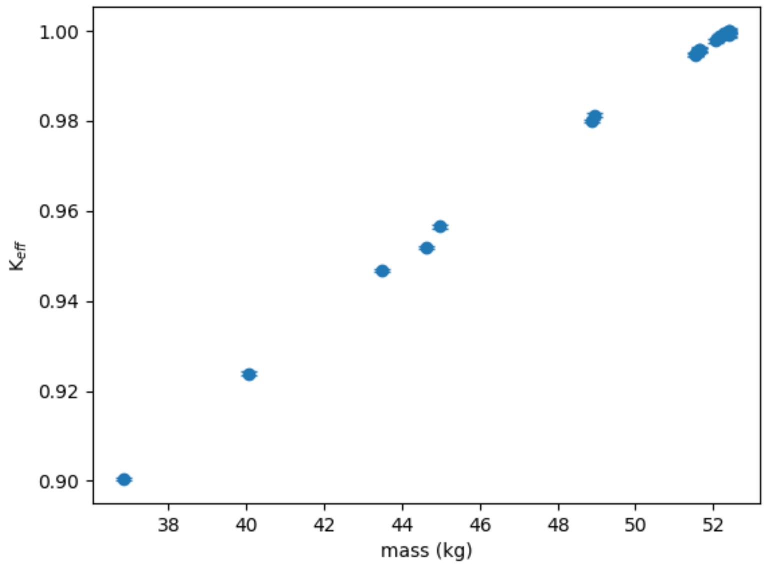


Figure 9.6: **The k$_{eff}$ vs mass for the UM Godiva spheres**. Note that the number of starting particles (10^8) was large enough to reduce the error bars significantly. The data for this plot are from Table 9.1.

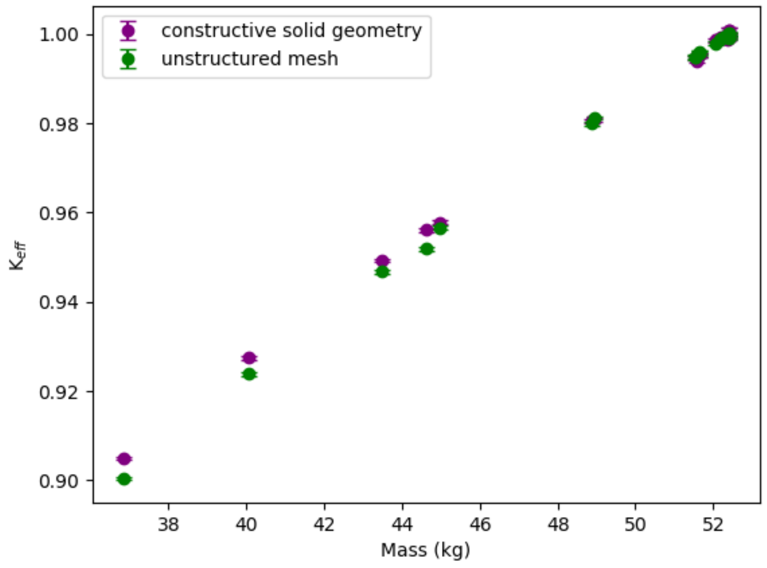
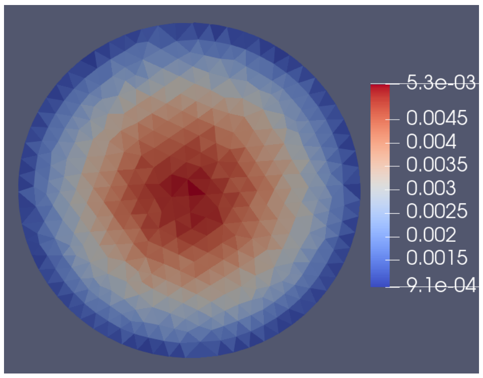
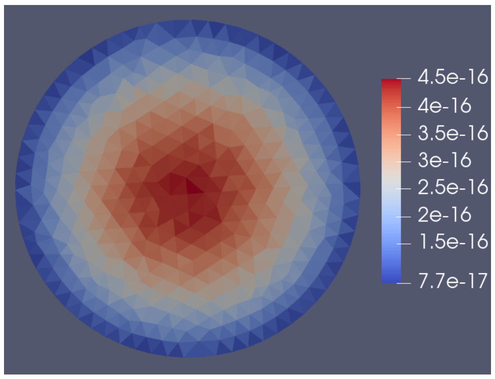


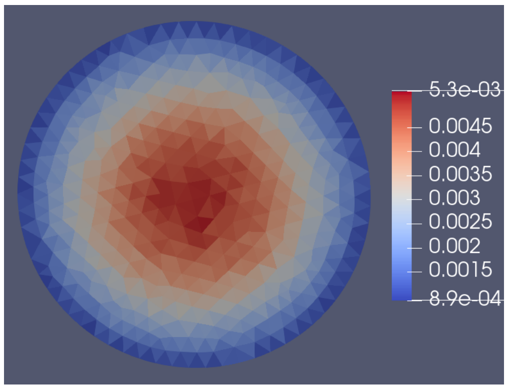
Figure 9.7: **A comparison of k$_{eff}$ vs mass for UM and CSG for the Godiva spheres**. Note that the number of particle histories was large enough to reduce the error bars significantly. The data for this figure are from Table 9.2



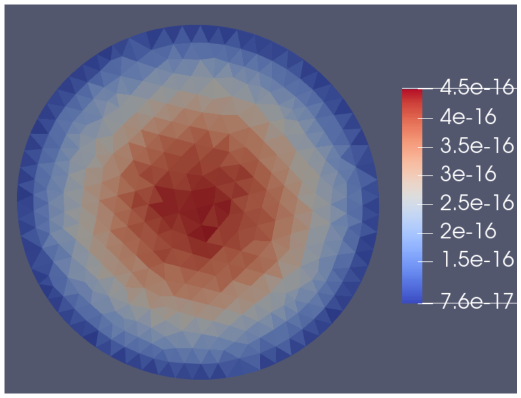
(a) Neutron flux (particles/cm^2) - 35453 1st order tet elements



(b) Neutron and photon energy deposition (MeV/g) - 35453 1st order tet elements

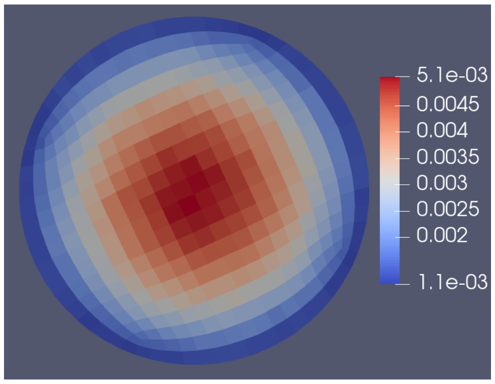


(c) Neutron flux (particles/cm^2) - 35453 2nd order tet elements

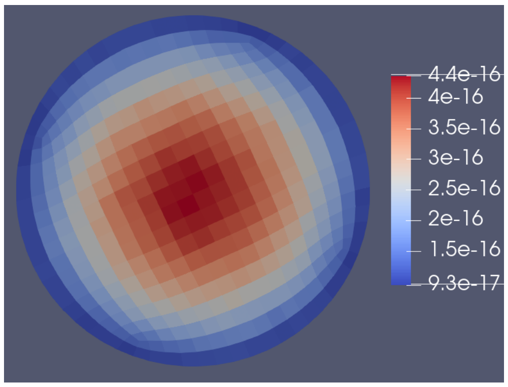


(d) Neutron and photon energy deposition (MeV/g) - 35453 2nd order tet elements

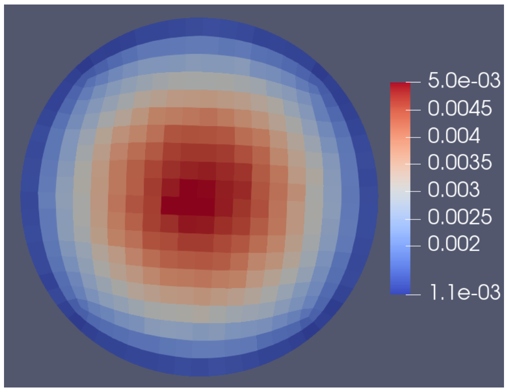
Figure 9.8: **Comparison of tallies for Godiva spheres with various tet meshes**. Each tally is normalized to the number of histories in the run (2×10^6). The quantities for the colorbar of each figure is listed in their respective captions. These meshes used here all have a k_{eff} value near 1.



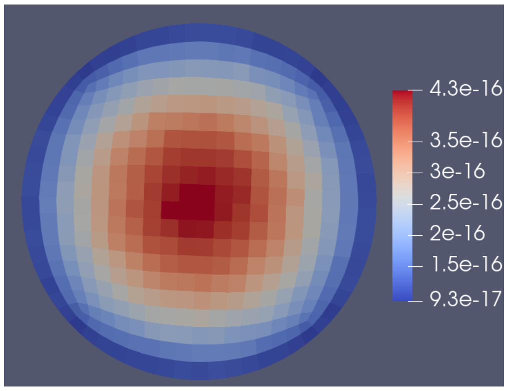
(a) Neutron flux (particles/cm^2) - 7168 1st order hex elements



(b) Neutron and photon energy deposition (MeV/g) - 7168 1st order hex elements



(c) Neutron flux (particles/cm^2) - 7168 2nd order hex elements



(d) Neutron and photon energy deposition (MeV/g) - 7168 2nd order hex elements

Figure 9.9: **Comparison of tallies for Godiva spheres with various hex meshes**. Each tally is normalized to the number of histories in the run (2×10^6). The quantities for the colorbar of each figure is listed in their respective captions. These meshes used here all have a k_{eff} value near 1.

experimental system. We are given the MCNP CSG input files.

9.7.2 Cubit meshing process

In order to recreate the sphere as an unstructured mesh, we used Cubit, a geometry and mesh generation toolkit developed by Sandia National Laboratory (`https://cubit.sandia.gov`). We used both version 15.3 and version 15.6. We used two different approaches to construct meshes of the same CSG models.

One can begin work in cubit by importing a geometry from many other programs, but we began creating our geometry using CSG techniques. Cubit provides all the necessary operations to create CSG geometries, so this was the natural process to use. Given the surfaces from the CSG model used for MCNP itself, and the aid of a plot from MCNP plot, the exact same surfaces could be used to recreate the same cells as volumes in Cubit. We noticed that there were two connected cells with the same material in the Oktavian geometry. In one of our approaches (hereon referred to as version 1), the cells were kept separate, as they were in the original file. In another approach (version 2), the cells that contained the same material were combined them into one volume.

With the geometry imported, the next steps were to create a mesh from the geometry. Cubit provides plenty of tools for this as well. For our results, we specified that the meshes be made using 1st order tetrahedron. Each element in this scheme has four faces and nodes placed only at the vertices. Then, while creating the unstructured mesh, Cubit will check to make sure that the mesh is of a high enough quality. If the mesh is not of high enough quality, it will continue, but issue a warning. In version 1, these warnings were solved by reducing the value of the auto factor setting. This caused the program to place more points closer together on the edges of the geometries for use in polygon creation, which allowed for smaller polygons, which therefore better captured the shape of the geometries and allowed for a higher quality mesh. In version 2, meanwhile, this problem was solved by changing the deviation angle of the mesh generation parameters, which is the angle between the tetrahedron face and the plane of the original surface. A smaller angle will typically create more elements that more closely approximate the original geometry.

With the geometry properly meshed, it is then exported as an Abaqus input formatted file. There are still some changes that need to be made at this stage, however. The default output of the Abaqus file defines the elsets (or the set of elements) in the *Element keyword line and includes several parameters for the material that MCNP does not use or recognize. As such, there is a little manual modification left. One must move the description of the elsets in the *elset keyword line in the part. Several material properties are also written into a file but MCNP uses only the material name (*Material) and density (*Density). Since MCNP ignores other material lines, we remove all the information about the information of the materials except density. We then add in material statistics for the elsets of different materials (see Abaqus Input files in the Appendix). We then use write_mcnp_m_input.py to generate the MCNP input file. Figures 9.12a and 9.12b show the difference in element volumes between versions 1 and 2. Figures 9.12c and 9.12d show perfect agreement between densities in various regions of the Oktavian sphere.

9.7.3 Simulation Setup

We ran simulations for eleven different filling materials in the Oktavian sphere: Al, Co, Cr, Cu, LiF, Mo, Mn, Si, Ti, W, and Zr. The Al, Co, Cr, Ti, and W versions are all originally of the same specified geometry, so we were able to use the same mesh for each of these. Similarly, Cu, LiF, Mn, and Zr were the same geometry and shared a mesh. Mo and Si both required their own meshes as the original experiments for these materials had unique geometries. For this report, the aluminum assembly and the copper assembly will serve as the archetypes for the two primary Oktavian assemblies that were tested. For the four different geometries and two mesh versions per geometry, there were some small difference in volume of the meshes and of their original CSG counterparts. Across the iterations, the percent difference in volume was all less than 1%, with Version 2 of Silicon being an outlier at 3.6% (Table 9.4).

In the experimental Oktavian Benchmark, the filling material of the sphere was powdered. In our simulations, we accounted for this and use the density of the filling to values from the Oktavian Benchmark. Furthermore, each assembly was given a slightly difference energy distribution for the source neutrons at the center of the assembly. Figures 9.10 and 9.11 show the energy distribution for the source neutrons for the aluminum and copper assembly, respectively. We used an energy cutoff to terminate tracks for neutrons that are below 10 keV. Neutrons below that range no longer contribute meaningful interactions. One last specification is that, similar to the original CSG model, this simulation uses the MCNP SDEF card for neutron sources. This source was considered to be a reasonable approximation of what the actual physical source was.

Despite the different approaches between Version 1 and Version 2 to refine the mesh, the two version have very similar meshes. Figures 9.12a and 9.12b show that the element sizes are nearly the same. The largest elements from each version are the same volume, and the smallest element from each are within an order of magnitude from each other. In both version, the volume difference between the smallest and largest elements span several orders of magnitude. Figures 9.12c and 9.12d show that both versions have matching densities that correspond to the original CSG values.

9.7.4 Results

In order to compare the leakage from the UM and CSG versions, we placed a spherical surface around the Oktavian assembly to measure the surface current. The result for neutrons in version 2 can be seen in figures 9.13 and 9.15. These again match with our predictions, as the leakage distribution matches the neutron source distribution. The photon current leakage for version 2, meanwhile, can be seen in figures 9.14 and 9.16. Here, we can see that most of the photons that escape are at a lower energy. The most important aspect, however, is the overlap of the two data sets. Though the agreement isn't perfect, it is so close that the data overlap within the resolution of the logarithmically scaled graph at most points. In addition, table 9.3 shows the mean values for the CSG version as well as both of the UM versions. Here, we can see that the difference between the values are always within 1% of each other. Thus, we have verified agreement between the CSG and UM models.

With the unstructured mesh, we were able to tally surface flux and energy deposition on individual elements of the mesh. The results for the case of the aluminum sphere can be seen in Figures 9.17 and 9.18. These results were in line with what we would expect. Aluminum is not

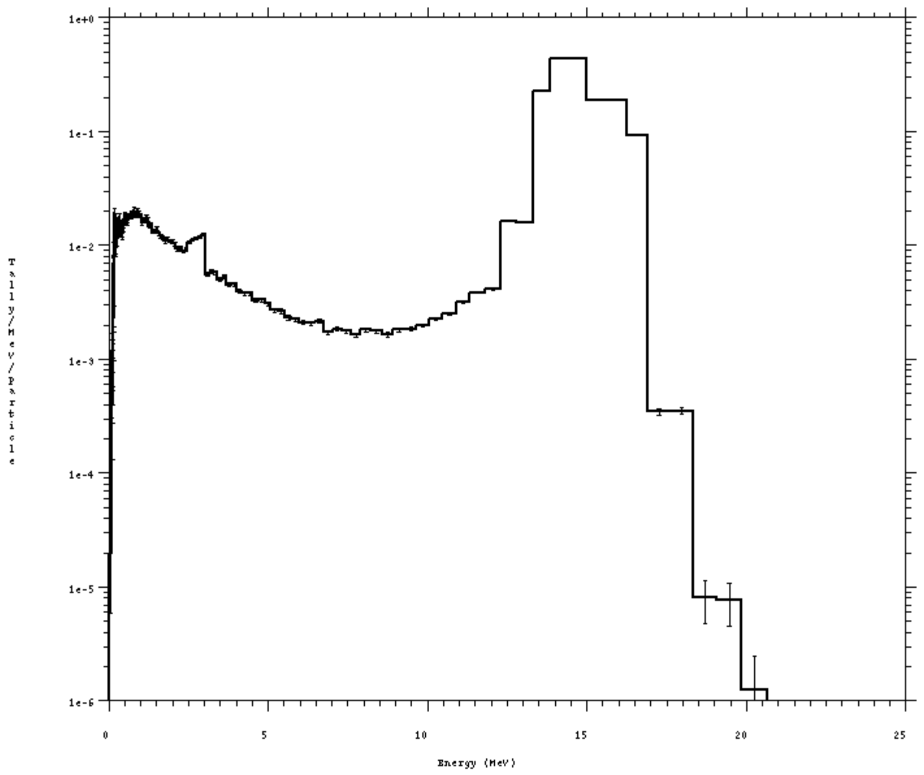


Figure 9.10: **Neutron source current for the aluminum Oktavian sphere**. The histogram values best represent a probability for a neutron of that specific energy being emitted from the source.

a good neutron refactor so most neutrons are leaked out. This is mirrored in Figures 9.17a and 9.17b of neutron flux, as well as 9.17c and 9.17d of neutron energy deposition. In these figures, the results are very high at the neutron source, but there is essentially no interaction throughout the rest of the system. We do, however, expect there to be many photon interactions, especially when source particles have reached the iron and aluminum to interact with. This is again mirrored in Figures 9.18a and 9.18b of neutron flux, as well as 9.18c and 9.18d of neutron energy deposition. There are relatively fewer interactions near the center, but once they the iron shell and aluminum is reached there are many reactions that decay as they move towards the edge of the system. Thus, the physics of the system acts as we would expect. Similar results can be seen in Figure 9.19, where the neutrons again barely interact outside of the source and the photons interacting strongly when they first come into contact with the material before decaying off.

9.8 Conclusion

In this project, we verified 3 models, namely a simple cube, the Godiva sphere, and the Oktavian sphere. Each of these models showed the agreement that we expected. The simple cube, a CSG model that could be perfectly represented as an UM model, showed perfect agreement between the CSG and UM calculation results. This result gave us confidence that UM and CSG calculations

		CSG Mean	V1 UM Mean	V2 UM Mean
Aluminum	N-current	0.9655 ± 10^{-5}	0.9656 ± 10^{-5}	0.9649 ± 10^{-5}
	P-current	$0.3976 \pm 8 \cdot 10^{-5}$	$0.3965 \pm 8 \cdot 10^{-5}$	$0.3952 \pm 8 \cdot 10^{-5}$
Cobalt	N-current	1.1056 ± 10^{-5}	1.10513 ± 10^{-5}	1.1058 ± 10^{-5}
	P-current	0.4092 ± 10^{-4}	0.40949 $\pm 8 \cdot 10^{-5}$	0.4080 ± 10^{-4}
Chromium	N-current	1.0757 ± 10^{-5}	1.07544 ± 10^{-5}	1.0758 ± 10^{-5}
	P-current	0.6435 ± 10^{-4}	0.64511 ± 10^{-4}	0.6445 ± 10^{-4}
Cobalt	N-current	1.1056 ± 10^{-5}	1.10513 ± 10^{-5}	1.1058 ± 10^{-5}
	P-current	0.4092 ± 10^{-4}	0.40949 $\pm 8 \cdot 10^{-5}$	0.4080 ± 10^{-4}
Copper	N-current	1.1818 ± 10^{-4}	1.18222 ± 10^{-4}	1.1824 ± 10^{-4}
	P-current	$0.2002 \pm 2 \cdot 10^{-4}$	$0.20113 \pm 2 \cdot 10^{-4}$	$0.2004 \pm 2 \cdot 10^{-4}$
Lithium Fluoride	N-current	0.8175 ± 10^{-5}	0.81866 ± 10^{-5}	0.8180 ± 10^{-5}
	P-current	0.4096 ± 10^{-4}	$0.40998 \pm 8 \cdot 10^{-5}$	0.4097 ± 10^{-4}
Manganese	N-current	1.3435 ± 10^{-4}	1.34329 ± 10^{-4}	1.3441 ± 10^{-4}
	P-current	$0.3039 \pm 2 \cdot 10^{-4}$	$0.30508 \pm 2 \cdot 10^{-4}$	$0.3046 \pm 2 \cdot 10^{-4}$
Molybdenum	N-current	1.2256 ± 10^{-5}	1.22538 ± 10^{-5}	1.2264 ± 10^{-5}
	P-current	-	-	-
Silicon	N-current	0.8430 ± 10^{-5}	0.83734 ± 10^{-5}	0.8343 ± 10^{-5}
	P-current	$0.6197 \pm 7 \cdot 10^{-5}$	$0.61421 \pm 6 \cdot 10^{-5}$	$0.6442 \pm 7 \cdot 10^{-5}$
Titanium	N-current	1.0576 ± 10^{-5}	1.05733 ± 10^{-5}	1.0575 ± 10^{-5}
	P-current	0.4314 ± 10^{-4}	$0.43110 \pm 9 \cdot 10^{-5}$	0.4298 ± 10^{-4}
Tungsten	N-current	1.2251 ± 10^{-5}	1.22408 ± 10^{-5}	1.2260 ± 10^{-5}
	P-current	$0.2186 \pm 6 \cdot 10^{-5}$	$0.21881 \pm 6 \cdot 10^{-5}$	$0.2177 \pm 6 \cdot 10^{-5}$
Zirconium	N-current	1.2971 ± 10^{-5}	1.29647 ± 10^{-5}	1.2974 ± 10^{-5}
	P-current	$0.3893 \pm 8 \cdot 10^{-5}$	-	$0.3895 \pm 8 \cdot 10^{-5}$

Table 9.3: **Surface current for the Oktavian models**. N-current is the neutron surface current and P-current is the photon surface current. The reported values represent the number of particles counted normalized to the number of source particles used for the run.

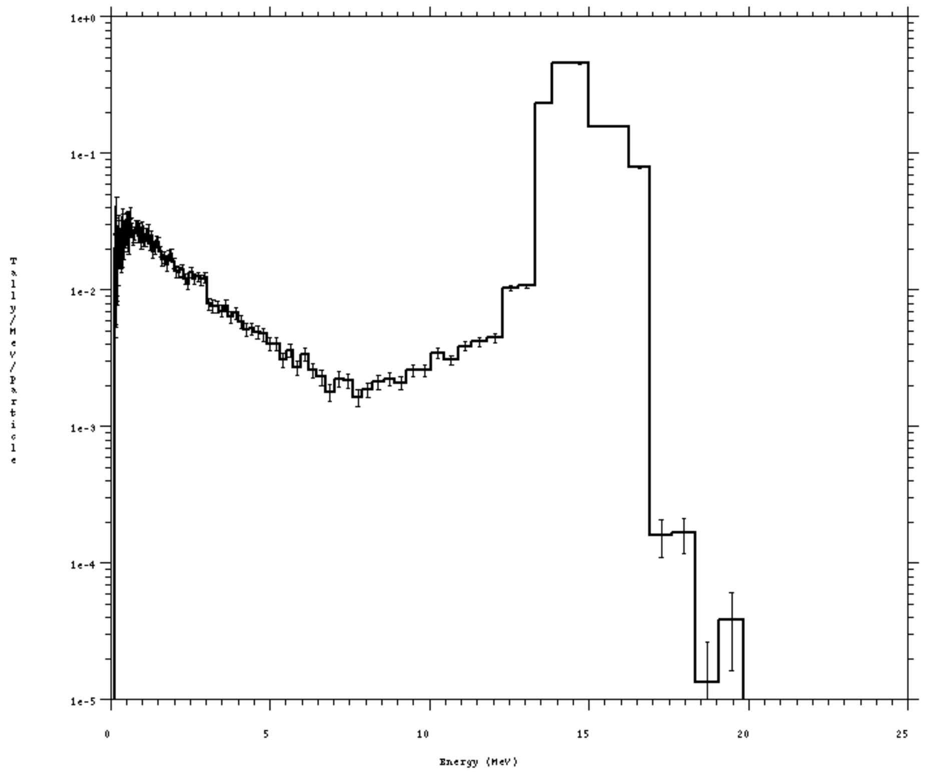
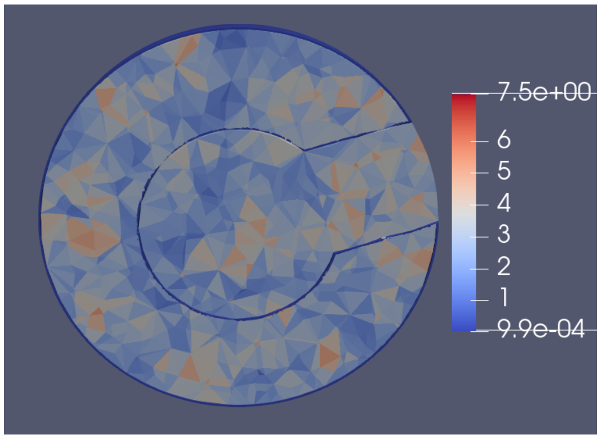


Figure 9.11: **Neutron source current for the copper Oktavian sphere**. The histogram values best represent a probability for a neutron of that specific energy being emitted from the source.

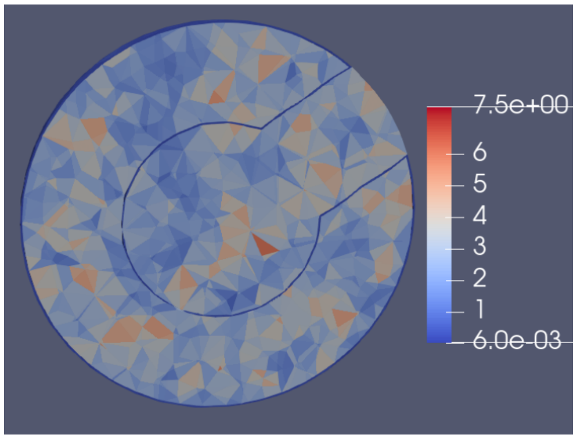
produce the physical effects. In other words, the small difference that we find between CSG and UM calculations is a result of a difference in a volume and geometry, not particle tracking methods implemented in the MCNP code. In the Godiva and Oktavian calculations, we saw agreement within

	CSG Volume (cm^3)	V1 Volume (cm^3)	V2 Volume (cm^3)
Aluminum	$3.326 \cdot 10^4$	$3.299 \cdot 10^4$	$3.298 \cdot 10^4$
Chromium	$3.326 \cdot 10^4$	$3.299 \cdot 10^4$	$3.298 \cdot 10^4$
Cobalt	$3.326 \cdot 10^4$	$3.299 \cdot 10^4$	$3.298 \cdot 10^4$
Copper	$1.188 \cdot 10^5$	$1.188 \cdot 10^5$	$1.184 \cdot 10^5$
Lithium Fluoride	$1.188 \cdot 10^5$	$1.188 \cdot 10^5$	$1.184 \cdot 10^5$
Manganese	$1.188 \cdot 10^5$	$1.188 \cdot 10^5$	$1.184 \cdot 10^5$
Molybdenum	$1.248 \cdot 10^5$	$1.248 \cdot 10^5$	$1.236 \cdot 10^5$
Silicon	$1.188 \cdot 10^5$	$1.178 \cdot 10^5$	$1.233 \cdot 10^5$
Titanium	$3.326 \cdot 10^4$	$3.299 \cdot 10^4$	$3.298 \cdot 10^4$
Tungsten	$3.326 \cdot 10^4$	$3.299 \cdot 10^4$	$3.298 \cdot 10^4$
Ziconium	$1.188 \cdot 10^5$	$1.188 \cdot 10^5$	$1.184 \cdot 10^5$

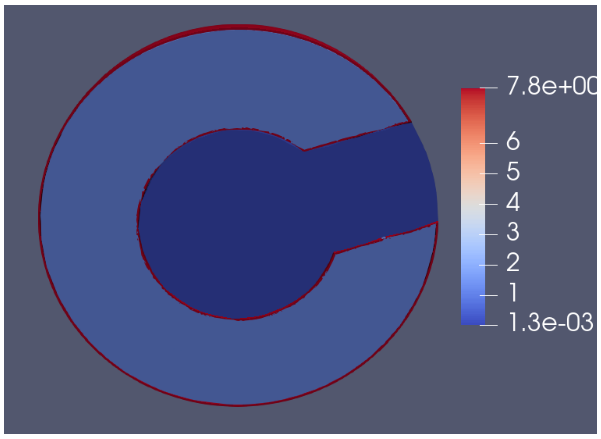
Table 9.4: **Volumes of the different versions of the Oktavian spheres.**



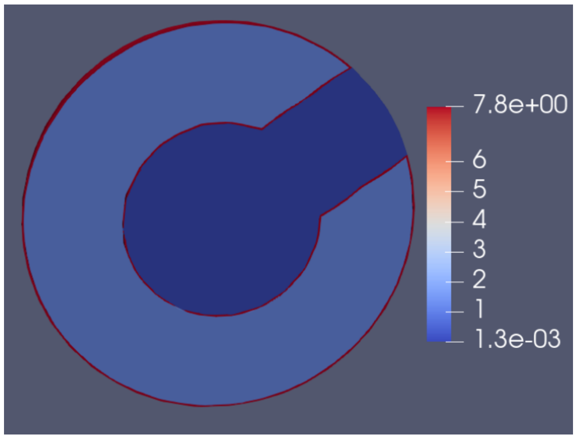
(a) Version 1 - Volume (cm^3)



(b) Version 2 - Volume (cm^3)



(c) Version 1 - Density (g/cm^3)



(d) Version 2 - Density (g/cm^3)

Figure 9.12: **Comparison of the volume and density of the elements from versions 1 and 2**. The units for each colorbar is listed in the respective caption.

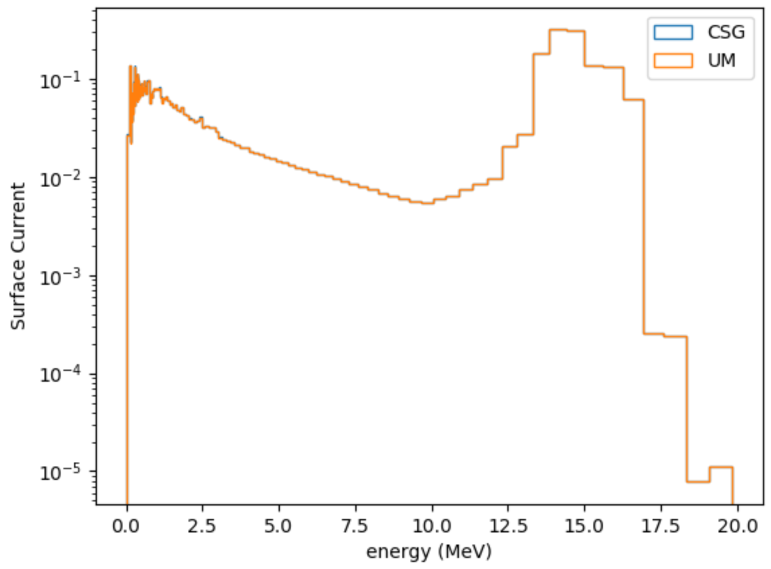


Figure 9.13: **Neutron surface current for the aluminum Oktavian sphere**. The UM data (in orange) are overplotted on the CSG data (in blue). With few and small exceptions, the UM data cover the CSG data. The histogram values here are normalized to the energy with of the bins.

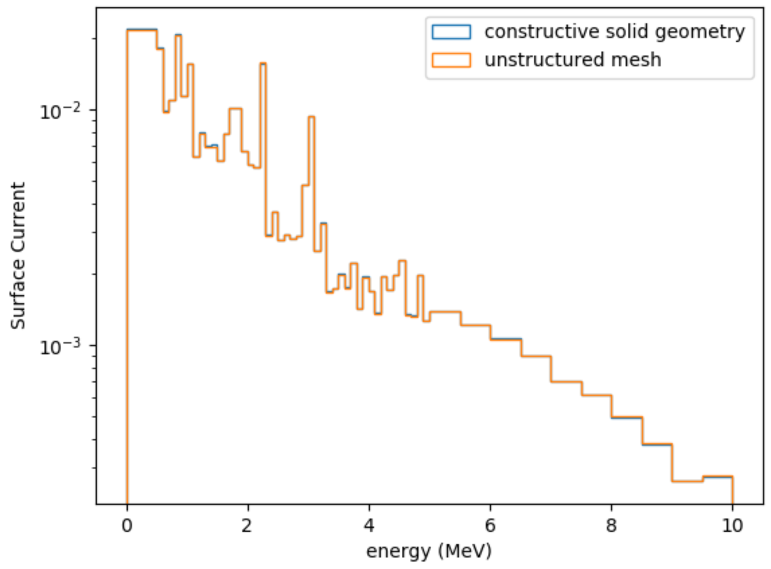


Figure 9.14: **Photon surface current for the aluminum Oktavian sphere**. With few and small exceptions, the UM data cover the CSG data. The histogram values are normalized to the energy with of the bins.

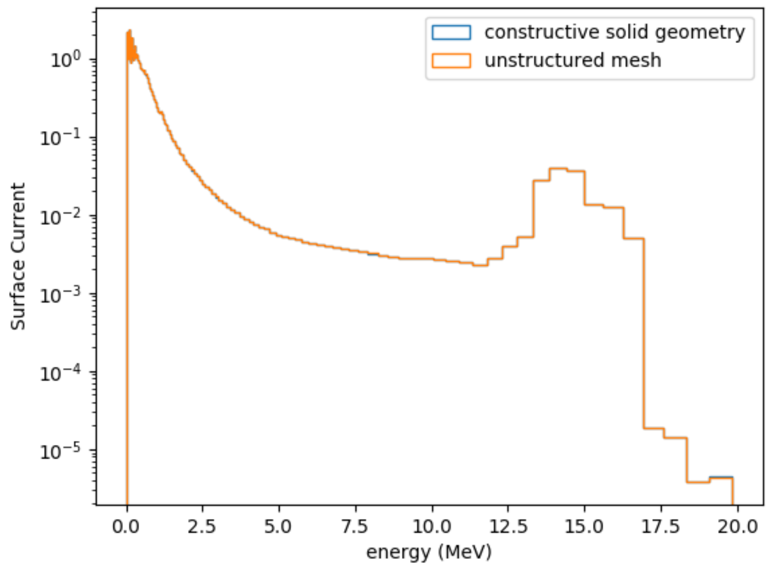


Figure 9.15: **Neutron surface current for the copper Oktavian sphere**. With few and small exceptions, the UM data cover the CSG data. The histogram values are normalized to the energy with of the bins.

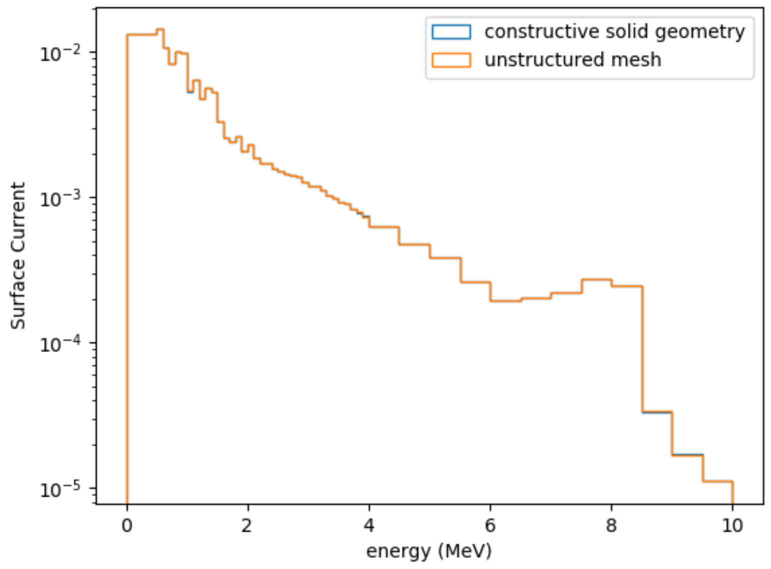
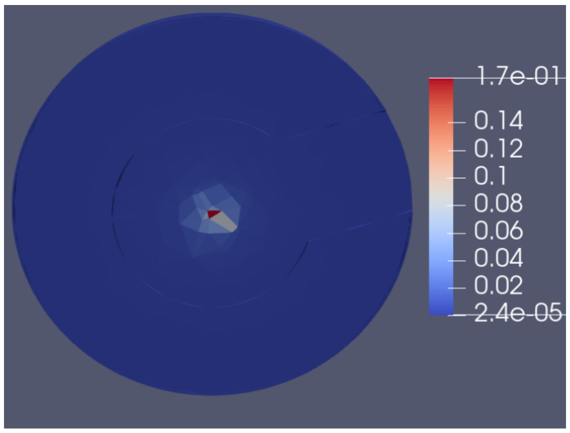
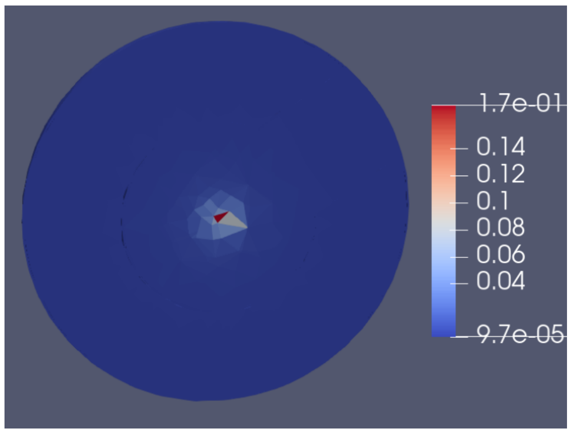


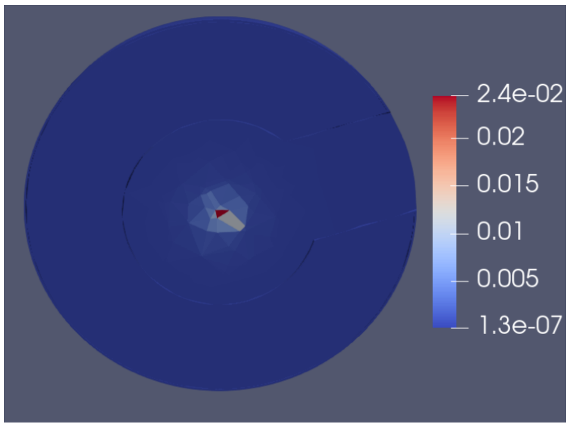
Figure 9.16: **Photon surface current for the copper Oktavian sphere**. With few and small exceptions, the UM data cover the CSG data. The histogram values are normalized to the energy with of the bins.



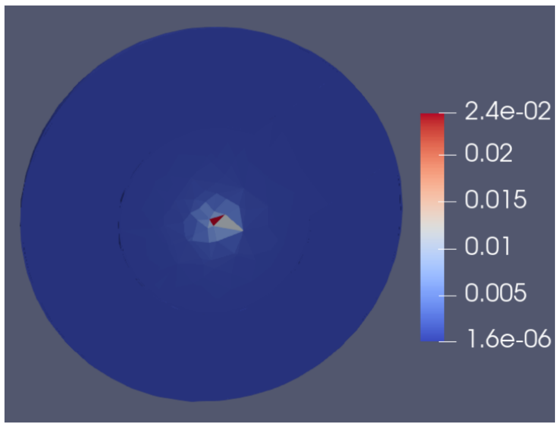
(a) Version 1 - neutron flux (particle/cm^2)



(b) Version 2 - neutron flux (particle/cm^2)

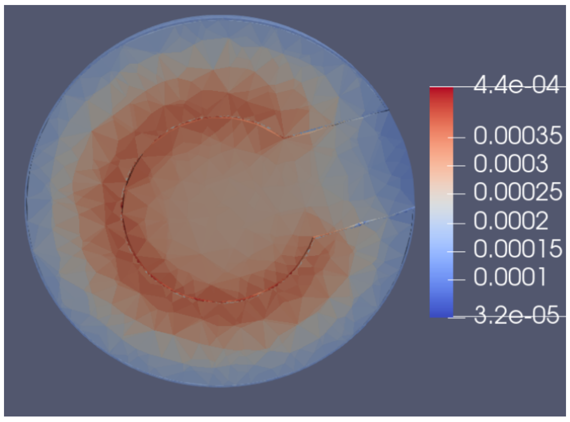


(c) V1 - neutron energy deposition (MeV/g)

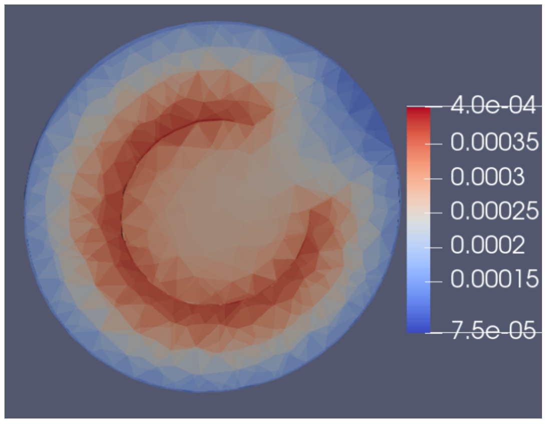


(d) V2 - neutron energy deposition (MeV/g)

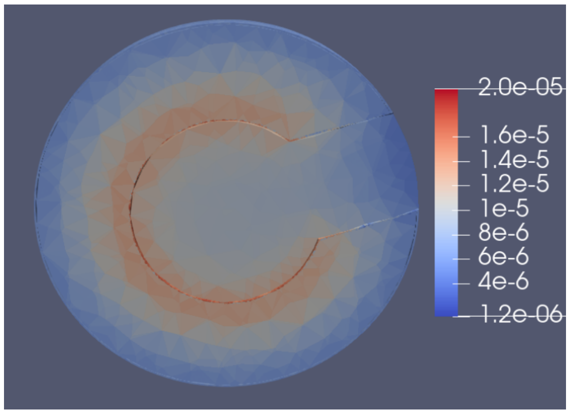
Figure 9.17: **Comparison of the neutron tallies from versions 1 and 2 of the Oktavian Sphere**.



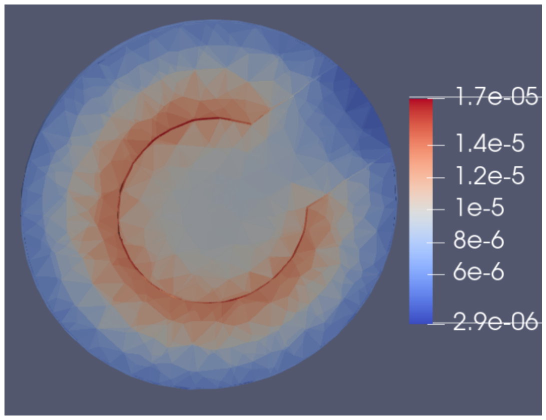
(a) Version 1 - photon flux (particle/cm^2)



(b) Version 2 - photon flux (particle/cm^2)

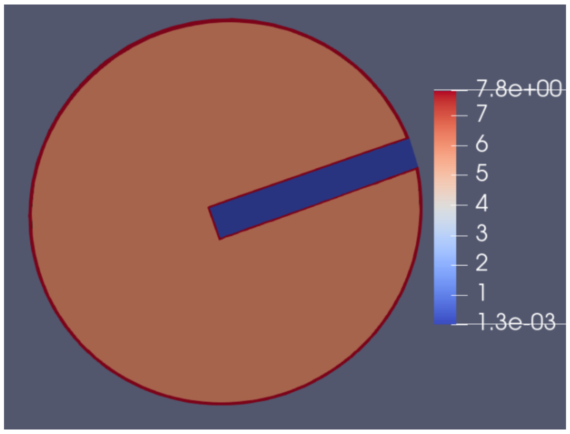


(c) V1 - photon energy deposition (MeV/g)

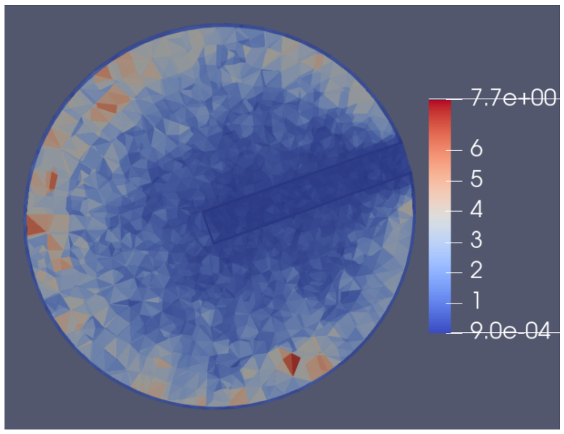


(d) V2 - photon energy deposition (MeV/g)

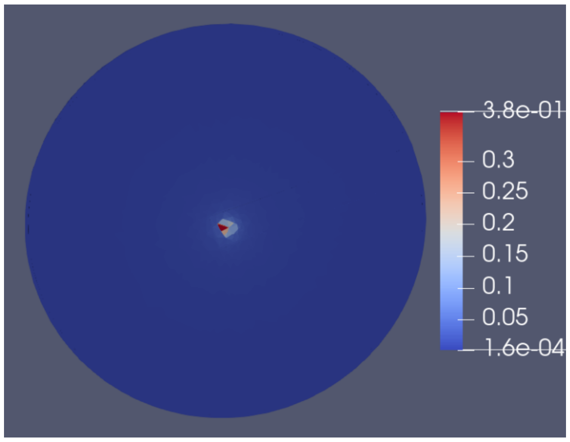
Figure 9.18: **Comparison of the photon tallies from versions 1 and 2 of the Oktavian Sphere**.



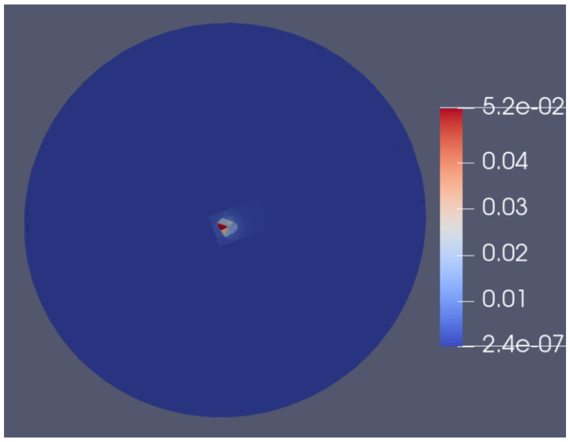
(a) Density (g/cm^3)



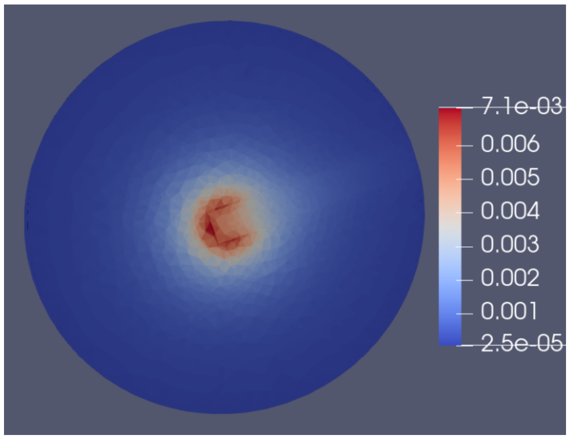
(b) Volume (cm^3)



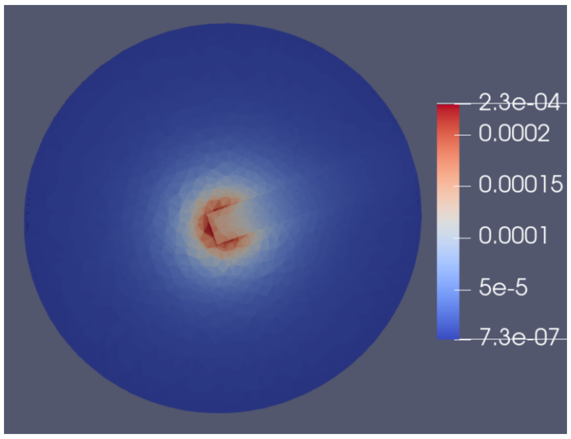
(c) Neutron flux (particles/cm^2)



(d) Neutron energy deposition (MeV/g)



(e) Photon flux (particle/cm^2)



(f) Photon energy deposition (MeV/g)

Figure 9.19: **Tallies from the copper Oktavian Sphere**. Subfigures 9.19c, 9.19d, 9.19e, and 9.19f are normalized to the nps (10^8 in this case).

a few percentage points, which was again as we expected. Thus, we conclude that we have verified that the UM and CSG calculations are consistent. In addition, we have expanded the verification suite for these assemblies that can be included in future MCNP code releases. With the growing use of MCNP UM simulations, this suite will help give users the same confidence in their results that the CSG suites provide in earlier MCNP versions.

9.9 Acknowledgements

We would like to thank Jerawan Armstrong (XCP-3) and Jim Ferguson (XCP-8) for their mentorship and helpful guidance during this project. We also want to thank Dan Israel, Garry Maskaly, Madison Andrews, and everyone else who made this XCP Summer Workshop possible. Finally, we want to acknowledge Karen Kelly (E13) for the provision of Abaqus input file for the Godiva sphere and Mike Rising (XCP-3) for the provision of the Oktavian SINBAD Benchmark documents.

__Harrison Leiendecker__ received his BS in physics from Clemson University in 2018. He is now entering his 3rd year as a PhD student at the University of Wyoming. His research focuses on modeling hydrodynamics and radiative transfer in protoplanetary disks. The goal of this project is to better understand the conditions of planet formation.

__Alex Warhover__ is in his final semester before receiving his BS in Physics and BS in Computer Science from the Missouri University of Science and Technology. His research in physics focuses on analyzing anomalous diffusion in the presence of an absorbing wall, while his research in computer science focuses on comparing design-centric and data-centric methods for cyber attack detection. He plans to obtain a semester long post-bac before continuing on to get a PhD in physics doing computational work.

Bibliography

Forrest Brown. Monte Carlo Techniques for Nuclear Systems - Theory Lectures. Technical Report LA-UR-16-29043, Los Alamos National Laboratory, 2016.

J. A. Kulesza and T. C. McClanahan. A Python Script to Convert MCNP Unstructured Mesh Elemental Edit Output Files to XML-based VTK Files. Technical Report LA-UR-19-20291 (rev.1), Los Alamos National Laboratory, 2019.

Roger L. Martz. Unstructured Mesh User's Startup Guide. Technical Report LA-UR-12-00795, Los Alamos National Laboratory, Los Alamos, NM, USA, February 2012. URL `http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-12-00795`.

Roger L. Martz and David L. Crane. The MCNP6 Book on Unstructured Mesh Geometry: Foundations. Technical Report LA-UR-12-25478, Los Alamos National Laboratory, Los Alamos, NM, USA, November 2012.

Alberto Milocco. The Quality Assessment of the Oktavian Benchmark Experiments. Technical Report IJS-DP-102.14, Jozef Stefan Institute, 2009.

H Paxton. History of Critical Experiments at Pajarito Site. Technical Report LA-9685-H, Los Alamos National Laboratory, 1983.

Christopher J. Werner, Jerawan Armstrong, Forrest B. Brown, Jeffrey S. Bull, Laura Casswell, Lawrence J. Cox, David Dixon, R. Arthur Forster, John T. Goorley, H. Grady Hughes, Jeffrey Favorite, Roger Martz, Stepan G. Mashnik, Michael E. Rising, Clell J. Solomon, Avneet Sood, Jeremy E. Sweezy, Anthony Zukaitis, Casey Anderson, Jay S. Elson, Joe W. Durkee, Russell C. Johns, Gregg W. McKinney, Garrett E. McMath, John S. Hendricks, Denise B. Pelowitz, Richard E. Prael, Thomas E. Booth, Michael R. James, Michael L. Fensin, Trevor A. Wilcox, and Brian C. Kiedrowski. MCNP User's Manual, Code Version 6.2. Technical Report LA-UR-17-29981, Los Alamos National Laboratory, Los Alamos, NM, USA, October 2017. URL `http://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-17-29981`.

9.A Codes

Cubit Journal File for Aluminum Oktavian Sphere Version 1

```
#!python
cubit.cmd('create sphere radius 10')
cubit.cmd('create cylinder height 20 radius 5.55')
cubit.cmd('from 100 50 100')
cubit.cmd('rotate volume 2 angle 90 about Y include_merged')
cubit.cmd('move volume 2 x 15 include_merged')
cubit.cmd('create sphere radius 19.95')
cubit.cmd('intersect volume 2 3')
cubit.cmd('unite volume 1 3')
cubit.cmd('create sphere radius 10.2')
cubit.cmd('create cylinder height 20 radius 5.75')
cubit.cmd('rotate volume 5 angle 90 about Y include_merged')
cubit.cmd('move volume 5 x 15 include_merged')
cubit.cmd('create sphere radius 19.95')
cubit.cmd('intersect volume 5 6')
cubit.cmd('unite volume 4 6')
cubit.cmd('subtract volume 1 from 4 keep')
cubit.cmd('delete volume 4')
cubit.cmd('create sphere radius 19.75')
cubit.cmd('subtract volume 1 7 from 8 keep')
cubit.cmd('delete volume 8')
cubit.cmd('create sphere radius 19.95')
cubit.cmd('subtract volume 1 7 9 from 10 keep')
cubit.cmd('delete volume 10')
cubit.cmd('volume 1 Scheme Tetmesh proximity layers off ')
cubit.cmd('mesh volume 1')
cubit.cmd('volume 7 Scheme Tetmesh proximity layers off ')
cubit.cmd('volume 7 size auto factor 4')
cubit.cmd('mesh volume 7')
cubit.cmd('volume 9 Scheme Tetmesh proximity layers off ')
cubit.cmd('mesh volume 9')
cubit.cmd('volume 11 Scheme Tetmesh proximity layers off ')
cubit.cmd('volume 11 size auto factor 4')
cubit.cmd('mesh volume 11')
```

Cubit Journal File for Aluminum Oktavian Sphere Version 2

```
#!python
cubit.cmd('create sphere radius 10')
cubit.cmd('create cylinder height 20 radius 5.55')
cubit.cmd('from 100 50 100')
cubit.cmd('rotate volume 2 angle 90 about Y include_merged')
cubit.cmd('move volume 2 x 15 include_merged')
cubit.cmd('create sphere radius 19.95')
cubit.cmd('intersect volume 2 3')
cubit.cmd('unite volume 1 3')
cubit.cmd('create sphere radius 10.2')
cubit.cmd('create cylinder height 20 radius 5.75')
cubit.cmd('rotate volume 5 angle 90 about Y include_merged')
cubit.cmd('move volume 5 x 15 include_merged')
cubit.cmd('create sphere radius 19.95')
cubit.cmd('intersect volume 5 6')
cubit.cmd('unite volume 4 6')
cubit.cmd('subtract volume 1 from 4 keep')
cubit.cmd('delete volume 4')
cubit.cmd('create sphere radius 19.75')
cubit.cmd('subtract volume 1 7 from 8 keep')
cubit.cmd('delete volume 8')
cubit.cmd('create sphere radius 19.95')
cubit.cmd('subtract volume 1 7 9 from 10 keep')
cubit.cmd('delete volume 10')
cubit.cmd('unite volume 11 7  ')
cubit.cmd('create material name \'steel\' density 7.824 ')
cubit.cmd('create material name \'aluminum\' density 1.223 ')
cubit.cmd('create material name \'air\' density 0.00128 ')
cubit.cmd('volume 1  Scheme Tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 1  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.4')
cubit.cmd('Trimesher volume gradation 1.4')
cubit.cmd('mesh volume 1 ')
cubit.cmd('volume 1  Scheme Tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 1  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.4')
cubit.cmd('Trimesher volume gradation 1.4')
cubit.cmd('volume 9  Scheme Tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 9  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.4')
cubit.cmd('Trimesher volume gradation 1.4')
cubit.cmd('mesh volume 9 ')
cubit.cmd('volume 12  Scheme Tetmesh proximity layers off geometry approximation
 angle 13.5 ')
cubit.cmd('volume 12  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('mesh volume 12 ')
```

Abaqus Input File for Aluminum Oktavian Sphere Version 1

```
*HEADING
cubit(/home/awarhover/Desktop/OktavianSphere.inp): 07/17/2020: 15:57:08
version: 15.3
**
********************************** P A R T S **********************************
*PART, NAME=Part-Default
**
********************************** N O D E S **********************************
*NODE, NSET=ALLNODES
       1,    1.916246e+01,    6.796790e-16,    5.550000e+00
       2,    1.916246e+01,   -3.032489e+00,    4.648281e+00
       ... 50115 more lines of this format ...
**
********************************** E L E M E N T S ****************************
*ELEMENT, TYPE=C3D4
       1,   24525,   38348,   38270,   38483
       2,   24525,   24692,   38348,   38483
       ... 154498 more lines of this format ...
**
********************************** P R O P E R T I E S ************************
** Cell 1: element number 84400-86393,  material 1
** Cell 2: element number 86394-143938, material 3
** Cell 3: element number 143929-15400, material 2
** Cell 4: element number 1-84399,      material 4
*ELSET, elset=Set-material-statistic-04, generate
 1,  84399, 1
*ELSET, elset=Set-material-statistic-01, generate
 84400,  86393, 1
*ELSET, elset=Set-statistic-02, generate
 86394, 143928, 1
*Elset, elset=Set-material-03, generate
 86394, 143928, 1
*ELSET, elset = Set-statistic-03, generate
 143929, 154500, 1
*Elset, elset=Set-material-02, generate
 143929, 154500, 1
**
```

```
*END PART
**
**
**
********************************* E N D   P A R T S **********************************
**
**
********************************* A S S E M B L Y ************************************
**
*ASSEMBLY, NAME=ASSEMBLY1
**
*INSTANCE, NAME=Part-Default_1, PART=Part-Default
*END INSTANCE
**
*END ASSEMBLY
**
**
**
*MATERIAL, NAME=air-material_01
*DENSITY
1.288000e-03
*MATERIAL, NAME=material_02
*DENSITY
1.223000e+00
*MATERIAL, NAME=material_03
*DENSITY
7.824000e+00
*MATERIAL, NAME=material_04
*DENSITY
7.8240000e+00
```

Abaqus Input File for Aluminum Oktavian Sphere Version 2

```
*HEADING
********************************* P A R T S *********************************
*PART, NAME=Part-Default
**
********************************* N O D E S *********************************
*NODE, NSET=ALLNODES
       1,    1.916246e+01,    6.796790e-16,    5.550000e+00
       2,    1.916246e+01,   -3.032489e+00,    4.648281e+00
       ... 4685 more lines of this format ...
**
********************************* E L E M E N T S ***************************
*ELEMENT, TYPE=C3D4
       1,    3581,    3572,    3550,    3629
       2,    3581,    3573,    3584,    4417
       ... 19155 more lines of this format ...
**
*ELSET, elset = Set-material-statistic_02, generate
 1, 6591, 1
*ELSET, elset = Set-material-statistic_01, generate
 8586, 19157, 1
*ELSET, elset = Set-material-statistic_03, generate
 6592, 8585, 1
********************************* P R O P E R T I E S ***********************
**
*END PART
**
**
**
********************************* E N D   P A R T S **************************
*******
**
**
********************************* A S S E M B L Y ****************************
```

```
*******
**
*ASSEMBLY, NAME=ASSEMBLY1
**
*INSTANCE, NAME=Part-Default_1, PART=Part-Default
*END INSTANCE
**
*END ASSEMBLY
**
**
**
*MATERIAL, NAME = material_01
*ELASTIC, TYPE=ISOTROPIC
0.000000e+00, 0.000000e+00
*DENSITY
7.824000e+00
*CONDUCTIVITY,TYPE=ISO
0.000000e+00
*SPECIFIC HEAT
0.000000e+00
**
*MATERIAL, NAME = material_02
*ELASTIC, TYPE=ISOTROPIC
0.000000e+00, 0.000000e+00
*DENSITY
1.223000e+00
*CONDUCTIVITY,TYPE=ISO
0.000000e+00
*SPECIFIC HEAT
0.000000e+00
**
*MATERIAL, NAME = material_03
*ELASTIC, TYPE=ISOTROPIC
0.000000e+00, 0.000000e+00
*DENSITY
1.280000e-03
*CONDUCTIVITY,TYPE=ISO
0.000000e+00
*SPECIFIC HEAT
0.000000e+00
**
************************************** H I S T O R Y ***************************
**********
**
*PREPRINT
**
**************************************** S T E P 1 ****************************
**********
*STEP,INC=100,NAME=Default Set
**
*STATIC
1, 1, 1e-05, 1
**
**
**
**
*END STEP
```

Cubit Journal File for Copper Oktavian Sphere Version 1

```
#!python
cubit.cmd('create Cylinder height 40 radius 2.55 ')
cubit.cmd('from 100 50 100')
cubit.cmd('rotate Volume 1  angle 90  about Y include_merged ')
cubit.cmd('move Volume 1 x 17.45 include_merged ')
cubit.cmd('create sphere radius 30.5 ')
cubit.cmd('intersect volume 1 2 ')
cubit.cmd('create Cylinder height 40 radius 2.85 ')
cubit.cmd('rotate Volume 3  angle 90  about Y include_merged ')
cubit.cmd('move Volume 3 x 17.15 include_merged ')
cubit.cmd('create sphere radius 30.0')
cubit.cmd('intersect volume 3 4 ')
cubit.cmd('subtract volume 2 from volume 4  keep')
cubit.cmd('delete volume 4')
cubit.cmd('create sphere radius 30 ')
cubit.cmd('subtract volume 2 5 from volume 6  keep')
cubit.cmd('delete volume 6')
cubit.cmd('create sphere radius 30.5 ')
cubit.cmd('subtract volume 2 5 7 from volume 8  keep')
cubit.cmd('delete volume 8')
cubit.cmd('volume 2  Scheme Tetmesh proximity layers off ')
cubit.cmd('volume 2 size auto factor 5')
cubit.cmd('mesh volume 2')
cubit.cmd('volume 5  Scheme Tetmesh proximity layers off ')
cubit.cmd('volume 5 size auto factor 5')
cubit.cmd('mesh volume 5')
cubit.cmd('volume 7  Scheme Tetmesh proximity layers off')
cubit.cmd('volume 7 size auto factor 5')
cubit.cmd('mesh volume 7')
cubit.cmd('volume 9 Scheme Tetmesh proximity layers off ')
cubit.cmd('volume 9 size auto factor 5')
cubit.cmd('mesh volume 9')
```

Cubit Journal File for Copper Oktavian Sphere Version 2

```
#!python
cubit.cmd('create sphere radius 30 ')
cubit.cmd('create sphere radius 30.5 inner radius 30 ')
cubit.cmd('create Cylinder height 40 radius 2.85 ')
cubit.cmd('rotate Volume 3  angle 90  about Y include_merged ')
cubit.cmd('move Volume 3 location 17.5 0 0 include_merged ')
cubit.cmd('subtract volume 3 from volume 1 2  ')
cubit.cmd('create Cylinder height 32.893 radius 2.85 ')
cubit.cmd('create Cylinder height 32.893 radius 2.55 ')
cubit.cmd('move Volume 5 location 0 0 0.3 include_merged ')
cubit.cmd('subtract volume 5 from volume 4 ')
cubit.cmd('rotate Volume 4  angle 90  about Y include_merged ')
cubit.cmd('move Volume 4  location 13.9465 0 0 include_merged ')
cubit.cmd('create Cylinder height 32.5932 radius 2.55 ')
cubit.cmd('rotate Volume 6  angle 90  about Y include_merged ')
cubit.cmd('move Volume 6 location 14.0966 0 0 include_merged ')
cubit.cmd('unite volume 2 4')
cubit.cmd('create material name \'steel\' density 7.824 ')
cubit.cmd('create material name \'copper\' density 6.0123 ')
cubit.cmd('create material name \'air\' density 0.00128 ')
cubit.cmd('volume 6  Scheme Tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 6  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.4')
cubit.cmd('Trimesher volume gradation 1.4')
cubit.cmd('mesh volume 6 ')
cubit.cmd('volume 1  Scheme Tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 1  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.4')
cubit.cmd('Trimesher volume gradation 1.4')
cubit.cmd('mesh volume 1 ')
cubit.cmd('volume 7  Scheme Tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 7  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('mesh volume 7 ')
```

Cubit Journal File for Silicon Oktavian Sphere Version 1

```
#!python
cubit.cmd('create sphere radius 10')
cubit.cmd('create cylinder height 60 radius 5.55')
cubit.cmd('from 100 50 100')
cubit.cmd('rotate volume 2 angle 90 about Y include_merged')
cubit.cmd('move volume 2 x 35 include_merged')
cubit.cmd('create sphere radius 30.5')
cubit.cmd('intersect volume 2 3')
cubit.cmd('unite volume 1 3')
cubit.cmd('create sphere radius 10.2')
cubit.cmd('create cylinder height 60 radius 5.75')
cubit.cmd('rotate volume 5 angle 90 about Y include_merged')
cubit.cmd('move volume 5 x 35 include_merged')
cubit.cmd('create sphere radius 30.5')
cubit.cmd('intersect volume 5 6')
cubit.cmd('unite volume 4 6')
cubit.cmd('subtract volume 1 from 4 keep')
cubit.cmd('delete volume 4')
cubit.cmd('create sphere radius 30.0')
cubit.cmd('subtract volume 1 7 from 8 keep')
cubit.cmd('delete volume 8')
cubit.cmd('create sphere radius 30.5')
cubit.cmd('subtract volume 1 7 9 from 10 keep')
cubit.cmd('delete volume 10')
cubit.cmd('volume 1 Scheme Tetmesh proximity layers off ')
cubit.cmd('mesh volume 1')
cubit.cmd('volume 7 Scheme Tetmesh proximity layers off ')
cubit.cmd('volume 7 size auto factor 4')
cubit.cmd('mesh volume 7')
cubit.cmd('volume 9 Scheme Tetmesh proximity layers off ')
cubit.cmd('mesh volume 9')
cubit.cmd('volume 11 Scheme Tetmesh proximity layers off ')
cubit.cmd('volume 11 size auto factor 4')
cubit.cmd('mesh volume 11')
```

Cubit Journal File for Silicon Oktavian Sphere Version 2

```
#!python
cubit.cmd('create sphere radius 10 ')
cubit.cmd('create sphere radius 10.2 inner radius 10 ')
cubit.cmd('create sphere radius 30.5 inner radius 10.2 ')
cubit.cmd('create sphere radius 30.5 inner radius 30 ')
cubit.cmd('create Cylinder height 40 radius 5.75 ')
cubit.cmd('rotate Volume 5 angle 90  about Y include_merged ')
cubit.cmd('move Volume 5 location 28.32 0 0 include_merged ')
cubit.cmd('subtract volume 5 from volume 3 4 2 1  ')
cubit.cmd('create Cylinder height 21.6331 radius 5.75 ')
cubit.cmd('create Cylinder height 22 radius 5.55 ')
cubit.cmd('subtract volume 7  from volume 6 ')
cubit.cmd('rotate Volume 6 angle 90  about Y include_merged ')
cubit.cmd('move Volume 6  location 19.13655 0 0 include_merged ')
cubit.cmd('create Cylinder height 21.6331 radius 5.55 ')
cubit.cmd('rotate Volume 8  angle 90  about Y include_merged ')
cubit.cmd('move Volume 8 location 19.13655 0 0 include_merged ')
cubit.cmd('unite volume 1 8  ')
cubit.cmd('unite volume 4 2 6  ')
cubit.cmd('create material name \'steel\' density 7.824 ')
cubit.cmd('create material name \'silicon\' density 1.29581 ')
cubit.cmd('create material name \'air\' density 0.00128 ')
cubit.cmd('volume 1  Scheme Tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 1  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('mesh volume 1 ')
cubit.cmd('volume 3 Scheme Tetmesh proximity layers off geometry approximation a
ngle 15 ')
cubit.cmd('volume 3 tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('mesh volume 3')
cubit.cmd('volume 9  Scheme Tetmesh proximity layers off geometry approximation
angle 12 ')
cubit.cmd('volume 9  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('mesh volume 9 ')
```

Cubit Journal File for Molybdenum Oktavian Sphere Version 1

```
#!python
cubit.cmd('create sphere radius 10')
cubit.cmd('create cylinder height 60 radius 2.55')
cubit.cmd('from 100 50 100')
cubit.cmd('rotate volume 2 angle 90 about Y include_merged')
cubit.cmd('move volume 2 x 35 include_merged')
cubit.cmd('create sphere radius 31.0')
cubit.cmd('intersect volume 2 3')
cubit.cmd('unite volume 1 3')
cubit.cmd('create sphere radius 10.2')
cubit.cmd('create cylinder height 60 radius 3.05')
cubit.cmd('rotate volume 5 angle 90 about Y include_merged')
cubit.cmd('move volume 5 x 35 include_merged')
cubit.cmd('create sphere radius 31.0')
cubit.cmd('intersect volume 5 6')
cubit.cmd('unite volume 4 6')
cubit.cmd('subtract volume 1 from 4 keep')
cubit.cmd('delete volume 4')
cubit.cmd('create sphere radius 30.5')
cubit.cmd('subtract volume 1 7 from 8 keep')
cubit.cmd('delete volume 8')
cubit.cmd('create sphere radius 31.0')
cubit.cmd('subtract volume 1 7 9 from 10 keep')
cubit.cmd('delete volume 10')
cubit.cmd('volume 1 Scheme Tetmesh proximity layers off ')
cubit.cmd('mesh volume 1')
cubit.cmd('volume 7 Scheme Tetmesh proximity layers off ')
cubit.cmd('volume 7 size auto factor 2')
cubit.cmd('mesh volume 7')
cubit.cmd('volume 9 Scheme Tetmesh proximity layers off ')
cubit.cmd('mesh volume 9')
cubit.cmd('volume 11 Scheme Tetmesh proximity layers off ')
cubit.cmd('volume 11 size auto factor 4')
cubit.cmd('mesh volume 11')
```

Cubit Journal File for Molybdenum Oktavian Sphere Version 2

```
#!python
cubit.cmd('create sphere radius 10 ')
cubit.cmd('create sphere radius 10.2 inner radius 10 ')
cubit.cmd('create sphere radius 30.5 inner radius 10.2 ')
cubit.cmd('brick x 70 y 70 z 70')
cubit.cmd('move Volume 4  location 43.32 0 0 include_merged ')
cubit.cmd('subtract volume 4 from volume 3 2 1 ')
cubit.cmd('create sphere radius 30.5 ')
cubit.cmd('brick x 70 y 70 z 70')
cubit.cmd('move Volume 6  location -26.68 0 0 include_merged ')
cubit.cmd('subtract volume 6 from volume 5 ')
cubit.cmd('unite volume 5 3  ')
cubit.cmd('create Cylinder height 3.05 radius  ')
cubit.cmd('create Cylinder height 30 radius 3.05 ')
cubit.cmd('rotate Volume 8  angle 90  about Y include_merged ')
cubit.cmd('move Volume 8 location 23.32 0 0 include_merged ')
cubit.cmd('create sphere radius 31 inner radius 30.5 ')
cubit.cmd('subtract volume 8 from volume 9 7  ')
cubit.cmd('create Cylinder height 22.5296 radius 3.05 ')
cubit.cmd('create Cylinder height 23 radius 2.55 ')
cubit.cmd('subtract volume 11  from volume 10  ')
cubit.cmd('rotate Volume 10 angle 90  about Y include_merged ')
cubit.cmd('move Volume 10 location 19.5848 0 0 include_merged ')
cubit.cmd('create Cylinder height 22.5296 radius 2.55 ')
cubit.cmd('rotate Volume 12  angle 90  about Y include_merged ')
cubit.cmd('move Volume 12 location 19.58479 0 0 include_merged ')
cubit.cmd('unite volume 1 12 ')
cubit.cmd('unite volume 9 10  ')
cubit.cmd('save as "/home/harrisonl/Desktop/cubit/mo/Mo_v1.cub" overwrite')
cubit.cmd('create material name \'steel\' density 7.824 ')
cubit.cmd('create material name \'molybdenum\' density 2.150 ')
cubit.cmd('create material name \'air\' density 0.00128 ')
cubit.cmd('volume 1 scheme tetmesh proximity layers off geometry approximation a
ngle 15 ')
cubit.cmd('volume 1 tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('Trimesher geometry sizing on')
cubit.cmd('volume 1 scheme tetmesh proximity layers off geometry approximation a
ngle 15 ')
cubit.cmd('volume 1 tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('Trimesher geometry sizing on')
cubit.cmd('mesh volume 1')
cubit.cmd('volume 2  scheme tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 2  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('Trimesher geometry sizing on')
cubit.cmd('volume 2  scheme tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 2  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('Trimesher geometry sizing on')
cubit.cmd('mesh volume 2 ')
cubit.cmd('volume 7  scheme tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 7  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('Trimesher geometry sizing on')
cubit.cmd('volume 7  scheme tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 7  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('Trimesher geometry sizing on')
cubit.cmd('mesh volume 7 ')
cubit.cmd('volume 9  scheme tetmesh proximity layers off geometry approximation
angle 15 ')
cubit.cmd('volume 9  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('Trimesher geometry sizing on')
cubit.cmd('volume 9  scheme tetmesh proximity layers off geometry approximation
angle 15 ')
```

```
cubit.cmd('volume 9  tetmesh growth_factor 1 ')
cubit.cmd('Trimesher surface gradation 1.3')
cubit.cmd('Trimesher volume gradation 1.3')
cubit.cmd('Trimesher geometry sizing on')
cubit.cmd('mesh volume 9 ')
```

Chapter 10

Novel numerical methods for turbulence

Team Members
Alex Somers and Francisco Holguin

Mentors
Sidharth GS and Gavin Portwood

Abstract

Two novel numerical strategies relevant to computation of turbulent flows are discussed. The first part focuses on the finite-scale equations that describe the evolution of finite volumes of fluid flow over time. Similar to Navier-Stokes, the finite-scale equations are continuum equations. However, the finite-scale equations contain additional momentum and internal energy transport terms. Stability properties of these equations due to these additional terms are not widely understood. This chapter outlines the efforts taken to understand the stability properties of the one-dimensional finite-scale convection-diffusion equation in order to better understand how the finite-scale terms affect the temporal stability of the finite-scale equations.

The second part focuses on scale-to-scale machine learning of poisson equation sources and their solutions, relevant to computation of incompressible turbulent flows. The ability of machine learning models to learn complex phenomena has been quickly improving over the last decade. The deep learning super resolution method is widely used to produce high resolution version of low resolution images. Super resolution could be used in order to improve numerical algorithms. Interpolation between different resolution grids is key part of the multigrid algorithm for efficiently solving elliptic partial differential equations. We apply the multigrid method to a particular example of the Poisson equation, the pressure-Poisson formulation of the two-dimensional Navier-Stokes equations. We also implement a super resolution generative-adversarial network (GAN) originally designed for images. We find issues in applying a relaxation operator at the coarsest levels in multigrid. We also find that the change from image data to pressure data requires more in-depth analysis of the GAN structure.

10.1 Stability Analysis of the Finite-Scale Convection Diffusion Equation

10.1.1 Introduction

The finite-scale equations (FSE), based on Burgers' equation and on incompressible Navier-Stokes equations, describe the evolution of finite volumes of fluid over time. The FSE are continuum equations derived from Boltzmann equations by averaging the velocity PDF in physical space. Similar to terms added in numerical simulation for high-speed flows (artificial viscosity) and for turbulent flows (subgrid scale models), the FSE contain additional momentum and internal energy transport terms L.G. Margolin (2020). These similarities suggest that the FSE may have implications in the field of numerical simulation of fluid flow L.G. Margolin (2002).

The scope of this project was to observe how the finite-scale observe length scale affects temporal stability in the FSE. It was determined that observing the finite-scale observe length scale term's effect on the temporal stability of the 1D finite-scale convection-diffusion equation would be an appropriate task for the ten week XCP Computational Physics Workshop (CPW). The overall goal was to compute and analyze the trends in the asymptotic instability of the equation.

10.1.2 Methodology

The FSE are typically unstable in computations. This has been well observed in large eddy simulation (LES) of turbulent flows and in the one dimensional counterpart to Burgers' equation, the finite-scale Burgers' equation Margolin (2009). The conservative finite-scale convection-diffusion equation contains each term in the traditional time-dependent convection-diffusion equation, but with an additional finite scale term:

$$\frac{\partial c}{\partial t} + \frac{\partial uc}{\partial x} = -L^2 \frac{\frac{\partial u}{\partial x}\frac{\partial c}{\partial x}}{\partial x} + \nu \frac{\partial^2 c}{\partial x^2} \tag{10.1}$$

In order to to better understand the stability properties of the FSE in general, we analyzed this equation, which is linear with respect to c. The goal was to compute and analyze the trends in the asymptotic instability of the equation. To accomplish this, the velocity field should be non-constant in order to activate the finite-scale term.

In order to observe the effects of the finite-scale term on stability, equation (1) was discretized in space using a fourth order accurate central differencing scheme. This produces a system of linear ordinary differential equations of the form:

$$C_t + \mathbf{A}C = 0 \tag{10.2}$$

where C is the vector of the solution variable at discrete points. For asymptotic stability of the system, solutions of the following form are sought:

$$C(t) = \hat{C}e^{\omega t} \tag{10.3}$$

This results in an eigenproblem of the following form:

$$(\omega + \mathbf{A})\hat{C} = 0 \tag{10.4}$$

Using a fourth order central differencing scheme with periodic boundary conditions and a constant velocity field with *u*=1, an eigenvalue spectrum can be generated from the coefficient matrix **A**. The real part of the eigenvalues can be seen below in Figure 1.

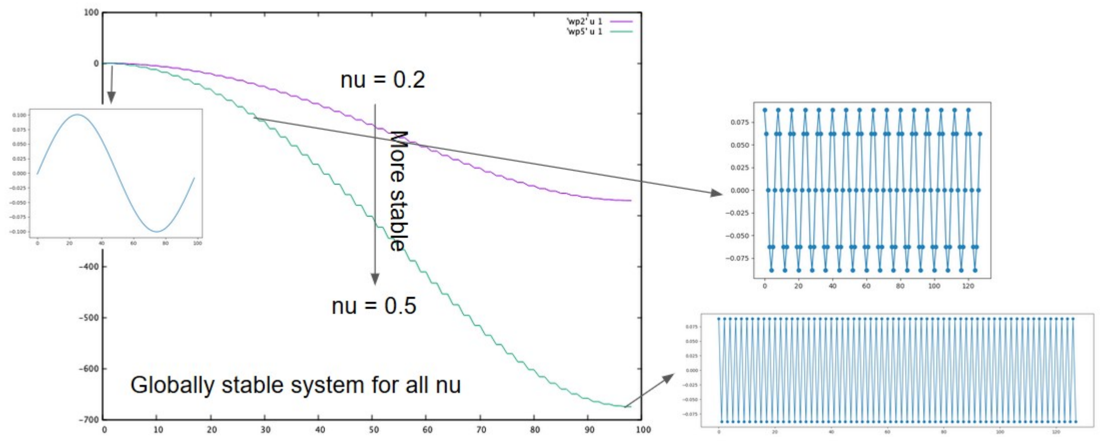


Figure 10.1: Real part of the eigenvalue spectrum for *u*=1

In Figure 1, we can see that all eigenvalues for this particular problem are non-positive. This is the condition necessary to ensure temporal stability for the system. Additionally, Figure 1 highlights that with increasing fluid viscosity, eigenvalues become more negative, showing that viscosity is a stabilizing factor in this problem. Spectra similar to this were generated to understand how different combinations of viscosity, velocity profile, and finite-scale term affect stability.

For this project, we applied four different types of velocity fields with periodic boundary conditions, all of which were constant with respect to time. First, we applied a constant velocity uniformly over the domain in order to verify correct implementation of the spatial discretization scheme. The eigenvalue spectrum for this case has been shown in Figure 1. In this case, the finite scale term is not activated. The subsequent three velocity profiles are shown in Figure 2 below. The constant frequency case was implemented such that the frequency of the velocity profile was variable. This made it possible to observe how an increase in frequency affected stability in tandem with viscosity and the finite-scale term. The last two spectra, LES and DNS spectra, were generated to simulate velocity profiles consistent with these two modeling approaches and provide insight as to how the finite-scale term affects stability in standard modeling scenarios.

In order to observe the effects of the finite-scale term on stability, a parametric study was performed in which ν and the velocity frequency, k_0, for various values of *L* in the FSCDE. Due to time constraints, L^2 was varied by order of magnitude from 10^-6 to 10^-1. This process resulted in an understanding of how these factors affect the stability of the FSCDE and subsequently an insight as to how the finite-scale term may affect the overall stability of the FSE.

10.1.3 Results

A parametric study was done to find critical ν values for which the **A** matrix produces non-positive eigenvalues. This process was performed keeping *L* constant and searching for the ν value at each

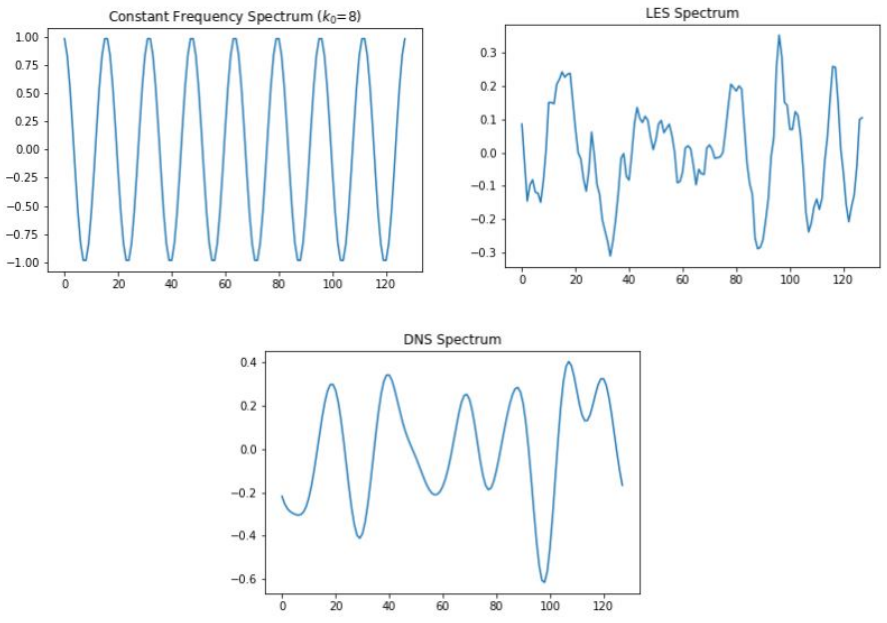


Figure 10.2: Velocity profiles used to observe stability of FSCDE (Top left: constant frequency sinusoidal profile, Top right: LES profile, Bottom: DNS profile)

k_0 where the maximum eigenvalue of $\mathbf{A}$ is non-positive. One such search for the critical ν value for a constant velocity field with k_0=32 and L^2=0.01 is shown in Figure 3 where the red dotted line is the largest eigenvalue found for each matrix generated for the corresponding ν. This process was

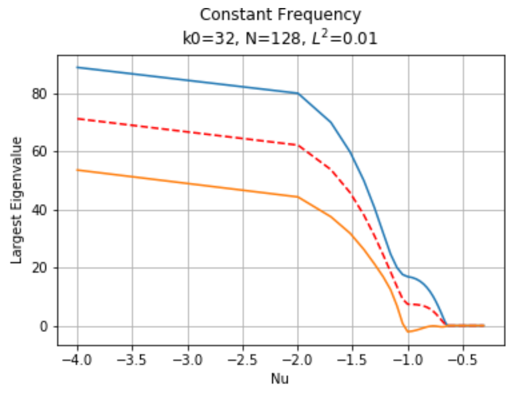


Figure 10.3: Critical ν search example

performed over six orders of magnitude in L^2 and frequencies from k_0=2 to k_0=32. It was found that not all cases converged monotonically to a stable matrix with increasing ν. In some cases, the system would become stable and then unstable with increasing ν multiple times before stability was reached. One example of this instability growth can be seen in Figure 4. While these instability growth instances occurred over many orders of magnitude (instability resurgence was observed on eigenvalue scales as small as 10^-12), the first instance of non-positive maximum eigenvalues was considere to be the critical ν value for which stability was observed.

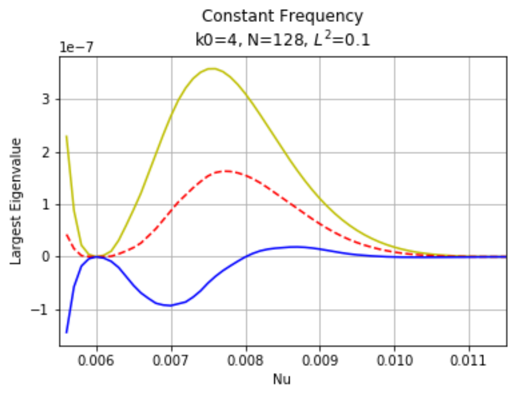


Figure 10.4: Instability growth with increasing ν for a constant frequency velocity field with k_0=4 and L^2=0.1

The finite scale term observer length scale (L^2) tends to be a destabilizing factor as it increases. It was observed that for increasing L there came a critical value for which the FSCDE becomes unstable. Beyond this critical L^2, no amount of physical viscosity can stabilize the system of equations. The general trend of increasing L is shown in Figure 5. It can be seen that spatial

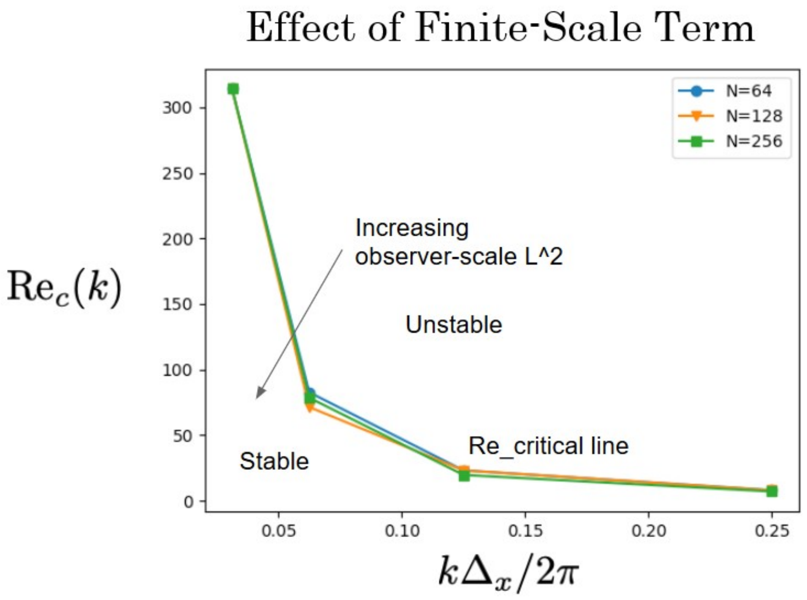


Figure 10.5: Effects of finite-scale term on stability for several spatial resolutions for three spatial resolutions (N=64, 128, 256)

resolution has little effect on the critical ν value (displayed as critical *Re* value). The general trend of growing instability with increasing L is still present regardless of spatial resolution. Additionally, as as L is increased, the area under the curve in Figure 5 becomes much smaller due to the increase in the eigenvalues produced for each k_0 and ν. Through these numerical experiments, the critical L

value approximately corresponds to the following:

$$L_{\text{crit}}k \approx \mathcal{O}(1) \tag{10.5}$$

Physically, this implies that the velocity-scale plays an important role in the stability of the finite-scale convection-diffusion equation.

10.1.4 Conclusion

The finite-scale convection-diffusion equations are shown to become increasingly unstable with increasing observer length-scale *L*. Beyond a certain critical observer length-scale value, L_{crit}, no amount of physical viscosity can stabilize the equations. For this idealized partial differential equation, stability is connected to the velocity length-scale.
Future work related to the stability properties of the FSE can focus on studying regularization strategies for the FSE. Additional work can also study the stability properties of the 1-D finite-scale convection-diffusion equation with time stepping and extending the equation to 2-D or even 3-D.
The author would like to thank mentors GS Sidharth and Gavin Portwood for their guidance over the course of the X-Division Computational Physics Summer Workshop.

10.2 Multigrid solvers with machine learning super resolution

10.2.1 Introduction

Partial differential equations describe the behavior of many physical phenomena, from the smallest quantum scales to the largest scales in the universe. Efficient numerical methods are needed to solve these equations, helping illuminate the complex natural world. Iterative methods, such as the Jacobi or Gauss-Seidel methods, are relatively simple algorithms which solve these equations accurately. However, the numerical solutions converge slowly using these methods as the high frequency errors are reduced quickly (smoothing), but the lower frequency errors remain (Trottenberg et al., 2001). The multigrid algorithm (e.g. Trottenberg et al., 2001) applies the smoothing operation across many grid length scales by repeatedly coarsening the grid and then interpolating back to the original high resolution grid. The result is that errors of all frequencies are reduced within a single iteration.

Often, a simple linear interpolation is used to change between grid resolutions. Machine learning could be used to improve this interpolation, making the multigrid method more efficient. Deep learning (LeCun and Hinton, 2015) methods have been able to capture the complexity of extremely complex systems, such as natural language or images. Super resolution methods have become adept at producing a realistic high resolution image from a low resolution image, even where the high resolution version does not actually exist (Yang et al., 2019). There are numerous methods of deep learning, one of which is a generative-adversarial network (GAN)(Goodfellow et al., 2014), which has become a popular choice for super resolution. GANs set up two independent models, the generator and the disriminator, which essentially compete against each other. The generator produces a low-resolution image from a high resolution image, learning from examples of low/high resolution pairs of training data. The discriminator receives a random low resolution image, either a real or generated, and determines its origin. The generator improves its ‘fake’ images in order to fool the discriminator and the discriminator improves its detection in order to not be fooled. Through this competition, the generator can learn to produce realistic high resolution images. The numerical grid is analogous to a photographic image. Super resolution could be used the in the interpolation steps between different grid resolutions.

Our goal is to use super resolution within the multigrid algorithm in order to solve the Poisson equation more efficiently. We train a GAN using high resolution and down-samped low resolution simulations. We then use the trained model in the coarsening or refining of the grid during a multigrid iteration. In this paper, we begin by describing the multigrid method and the specific physical data we use for the Poisson equation. We then describe the GAN network and the results of training. Finally, we compare the efficiency of multigrid in solving the Poisson equation between using traditional interpolation methods and using interpolation with GANs.

10.2.2 Methods

Poisson equation and numerics

We solve the Poisson equation in two dimensions, written as

$$\nabla^2 p = \frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = f(x, y) \tag{10.6}$$

where p is the solution variable of interest and f is a source function. We can solve the Poisson equation by setting up a diffusion equation

$$-\frac{\partial p}{\partial t} = \nabla^2 p - f(x, y) \tag{10.7}$$

and iterating over the time variable until convergence. We can discretize Eq.10.6 using a variety of schemes, such as forward time-centered space differencing $\nabla^2 p \approx (p^n_{i+1,j} + p^n_{i-1,j} + p^n_{i,j+1} + p^n_{i,j-1} - 4p^n_{i,j})$ where i and j are spatial indices and n is the time index. Eq.10.7 can be organized in the form Trottenberg et al. (2001)

$$p^{n+1}_{i,j} = p^n_{i,j} + \omega\ z^n_{i,j} \tag{10.8}$$

where z represents the discretized right hand side of Eq. 10.7 and ω is the relaxation parameter (i.e. time step). The choice of $\omega = 1$ results in the Jacobi (or equivalently the Gauss-Seidel) method

$$p^{n+1}_{i,j} = \frac{1}{4}(p^n_{i+1,j} + p^n_{i-1,j} + p^n_{i,j+1} + p^n_{i,j-1}) \tag{10.9}$$

The Gauss-Seidel method uses the values of p as they are updated, which results in a much faster convergence compared to the Jacobi method. Eq.10.9 shows that the Gauss-Seidel method is equivalent to updating a cell by a simple average of four surrounding cells. In other words, the Gauss-Seidel method smooths the grid: high frequency errors are reduced in a few iterations, but low frequency ones remain longer.

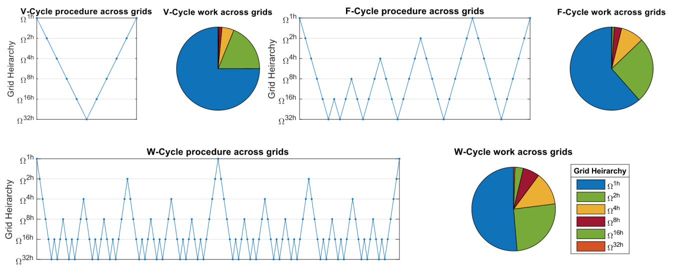


Figure 10.6: Diagram of three multigrid algorithms from Wikimedia commons (Adilnisar / CC BY-SA (https://creativecommons.org/licenses/by-sa/4.0). The top level is the grid at original resolution with progressively coarser resolution with lower levels.

Multigrid

The goal of the multigrid algorithm is to apply smoothing at multiple grid scales, as the high frequency errors (easily damped by Gauss-Seidel) on coarse grids are the same low frequency errors on the original, fine resolution grid. Multiple frequencies of error are efficiently reduced, resulting in much fewer iterations to convergence.

Generally, the algorithm consists of applying the Gauss-Seidel method (‘relaxation’), coarsening the grid, and then applying the Gauss-Seidel again, and so on, until a minimum resolution is reached, at which we interpolate to a finer grid and apply relaxation methods until the original grid resolution. Figure 10.6 shows a diagram of the three different multigrid cycles, where the top level represents the original resolution and the bottom level represents the coarsest resolution. The figure also shows the fraction of total work completed at each level.

Each iteration of multigrid requires many smoothing steps compared to only iteration, but the overall time to convergence is less. Figure 10.7 shows the scaling of the Gauss-Seidel and two multigrid methods with N the number of elements in the grid. The multigrid methods scale linearly, while the Gauss-Seidel method scales quadratically.

Super resolution: Generative adversarial network

Deep learning methods have been successfully applied in incredibly complex systems by employing the use of many interconnected neural network layers, often as a convolutional neural network. Even without any prior knowledge of the system, the network can learn relatively quickly. Super resolution refers to the application of these networks to low resolution images in order to predict the high resolution version. A generative adversarial network (GAN), introduced by Goodfellow et al. (2014), is particularly effective at super resolution.

We use the GAN from Birla (2018), the structure of which is shown in Figure 10.8. The image data is normalized to be between -1 and 1. The generator is made up of four components. The image is first passed through a convolutional layer. Second, there are four ‘residual blocks’ with each block consisting of convolution, batch normalization, and activation layer. The result of each block is summed with the grid preceding the block, in what is called a ‘skip connection.’ Third, there is an up-sampling layer, which increases the number of grid points in the image to match the higher-resolution desired. Finally, there is a convolutional layer which produces the final

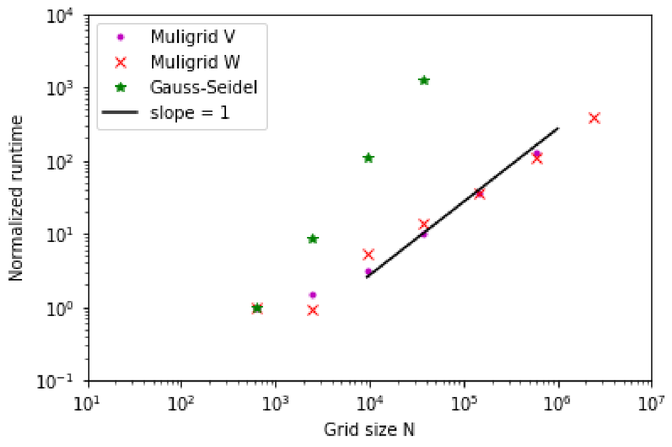


Figure 10.7: Scaling test for Gauss-Seidel, and multigrid V and W cycles. As expected, the multigrid cycles scale roughly $\propto N$, where N is the number of grid points. For the multigrid cycles, the coarsest grid is $N = 16$

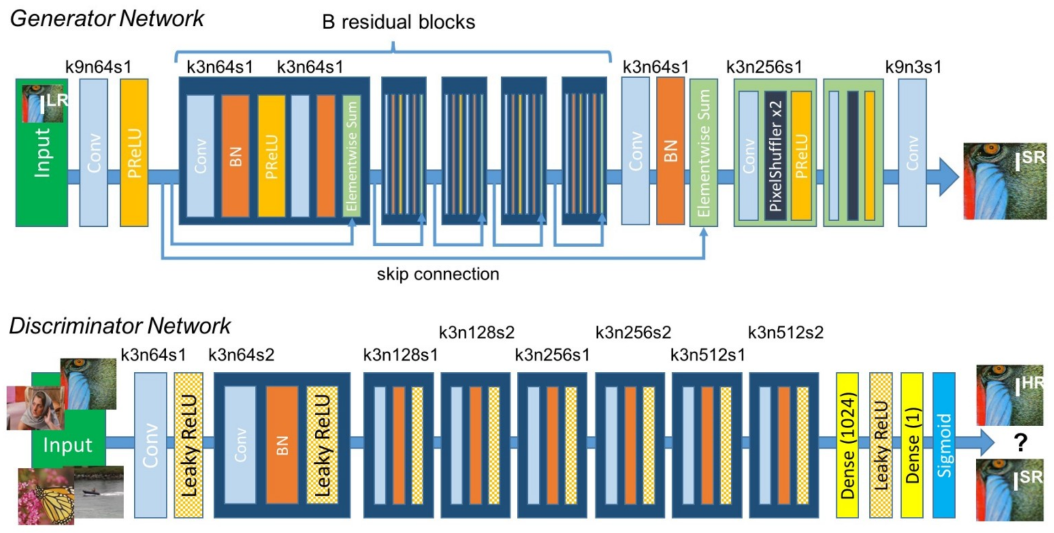


Figure 10.8: Overview of the GAN from Birla (2018), originally from Ledig et al. (2017), which we modified.

high-resolution image. The discriminator is simpler than the generator. It consists of a series of seven blocks containing convolution, batch normalization, and activation layers. The final sigmoid activation function produces a value for whether the input is from the real data set or the generator.

We modify the original GAN in order to better accommodate a grid of pressure values, instead of image values. Image grids typically contain three channels for each spatial two-dimensional coordinate, each channel representing RGB color. For simplicity, we construct an image grid by repeating the pressure value in each of the three channels. We also change the final activation layer from a ‘tanh’ function (which outputs from values from -1 to 1) to a ‘linear’ function that returns the value given.

10.2.3 Discussion

Solving the Poisson equation

The choice of system to solve is important, as choosing too simple of a system does not allow us to show differences between numerical methods. For example, we first solve the Laplace equation, where $f(x, y) = 0$ in Eq. 10.7, with all boundaries, except one, taking a value of 0. As Figure 10.7 shows, the multigrid method compared to Gauss-Seidel scales much better with increasing number of grid elements. Indeed, for this Laplace equation set up and a grid of 97^2, Gauss-Seidel converged in approxmiately 1000 steps, while multigrid converged in a single iteration. We can reduce the effectiveness of the multigrid method by increasing the definition of the lowest resolution grid, and increase the number of multigrid iterations to converge. Nonetheless, it is probably better to choose a more difficult set up from the beginning.

The pressure-Poisson formulation of the incompressible Navier-Stokes equations is one such physical system in the form of Eq. 10.7. The pressure-poisson equation is derived from taking the divergence of the fluid momentum conservation equation and rearranging to obtain the equation,

$$\nabla^2 p = \nabla \cdot (\nu \nabla^2 u - (u \cdot \nabla) u) = f(x, y), \tag{10.10}$$

where ν is the viscosity, u is the velocity, and p is the pressure. We set $\nu = 0$ for simplicity. The resulting system involves calculating the source term $f(x, y)$ from the velocity field u^n, solving the Poisson equation for pressure, and then using that pressure to update the velocity field u^{n+1}. In order to test the multigrid method and produce training data, we solve the two dimensional Navier-Stokes equation and randomly drive a velocity field within a periodic domain. We know the expected, true pressure solution, and the velocity field. Solving Eq. 10.10 to find the pressure is much more challenging than the Laplace equation, as $f(x, y)$ is determined by a non-linear dependence on the multiscale velocity field. Additionally, within the multigrid algorithm the $f(x, y)$ grid must also be interpolated or down-sampled alongside the pressure grid.

Applying the Gauss-Seidel for the Poisson equation over a large grid requires an order of magnitude more iterations to converge compared to the Laplace equation, but it does converge. Multigrid is more challenging. We found an issue with the coarest grids (13^2 or smaller). When applying iterative relaxation on those grids, the results converge to a different solution compared to the true value. This issue propagates back to the higher resolution grids during a multigrid step. An investigation into this behavior is ongoing. For now, we restrict the coarsest level to be greater than 13^2.

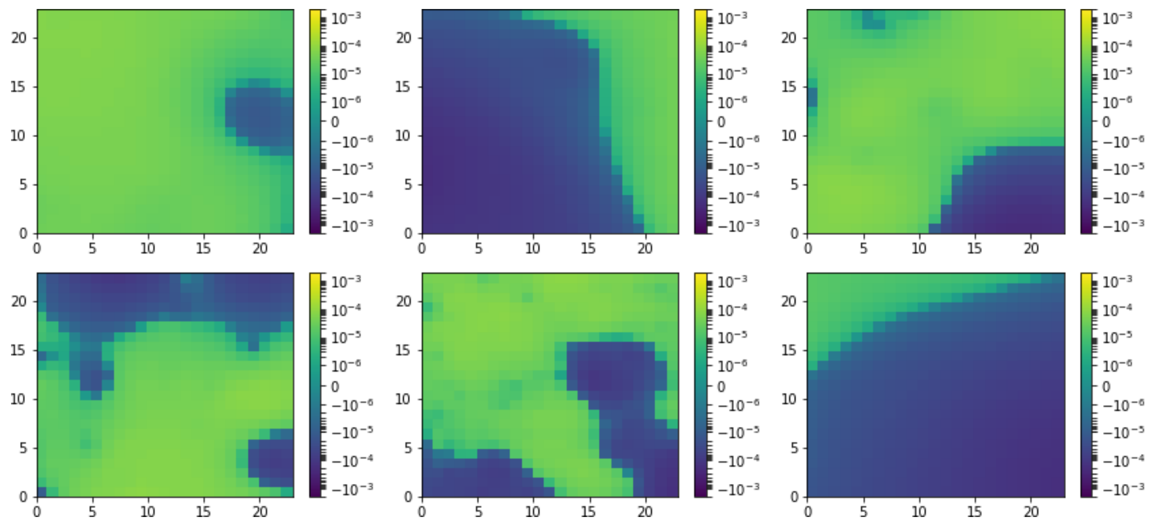


Figure 10.9: Training data examples: Random 24^2 image sections from all multigrid levels (where the original image is $192^2, 96^2, 48^2$ or 24^2).

Training

We produce 200 different examples of a randomly driven velocity field within a periodic domain of size 192^2, each with a source term $f(x, y)$ grid and pressure solution grid p. We will use the true pressure grid in order to train the GAN (Fig. 10.9). Training large grids is often difficult as it requires extremely large data sets and long training times. In order to streamline our training, we aim to train on smaller sections of the grid. We divide all of the grids at each multigrid level into 24^2 patches. The highest resolution grid of size 192^2 is split up into 8^2 patches. The next coarsest grid of size 96^2 is split up into 4^2 patches, and so on for coarser grids, until the 24^2 grid. The total number of grids available for training is thus $200 * (8^2 + 4^2 + 2^2 + 1^2) = 17000$ individual grids. It is also possible to introduce small random perturbations onto the images and use them to train the GAN. For example, the GAN should produce similar results for any grid translation in either spatial dimension. These translation greatly expand the space of training data. Even drawing random translations from -5 to 5 cell shifts in both dimensions would increase the training data space by a factor of 100.

For now, we explore training the GAN without perturbed grids, as it is a challenge to train the GAN on the pressure data. The original GAN implemented 'feature loss' for the generator, meaning that the loss in the generator was relative to the features of a pre-trained model ('VGG19'), which was trained on hundreds of thousands of images. This means that the generator could more quickly learn typical image features. Since those image features do not apply to the the fluid problem we consider, we change the loss to simple mean-squared error ('MSE'). Since we are not using pre-trained losses, compared to training the GAN on images, training the GAN on pressure data should be slower. Additionally, because the image data has values restricted between 0 and 255, it is easy to normalize the entire data set. Normalization around a mean of zero and a variance of unity greatly improves the ability of the GAN to learn. On the other hand, we could also benefit from the three RGB channels in the GAN. The GAN learns different features from each color channel in the image. We can pick the pressure and source data to represent a channel each, in order for the GAN to learn not only the relationship between high and low resolution data, but also the relationship between the source data and the pressure solution.

Current training of the GAN has been unsuccessful: the GAN has not learned any features of the data. It is unclear whether the issue is training time, training data set, GAN structure, or GAN hyperparameters. First, we expect the training to be slower, but it is unclear how much slower. Second, we produce a space of training data of $17,000$ grids (and train on a subset) that should be sufficient for training. Third, we changed several components in the GAN, such as the final activation layer in the generator or the data normalization. It is possible that more changes are needed in order for the GAN to better capture the pressure data. Fourth, the GAN hyperparameters control the efficiency of training. These hyperparameters include parameters such as number of training images, batch size, optimizer settings, and so on. We choose typical values for the hyperparameters, but often these values need to be tuned for each specific set up.

Multigrid and GAN

Once we have a trained the GAN to produce sufficiently accurate predictions of high resolution grids from low resolution grids, we will integrate the generator into the multigrid interpolation steps, both for coarsening and refining the grids.

10.2.4 Conclusion

We investigate the potential use of deep learning super resolution to improve the efficiency the multigrid algorithm. We identify the Poisson-pressure equation as the physical system to solve with multigrid. We implement the GAN from Ledig et al. (2017) and explore the possible changes needed in order to generalize it beyond only input images. We find the following:

1. Further work is needed in order to characterize the issue we found of applying Gauss-Seidel to very coarse grids. We can restrict the depth of the multigrid algorithm and avoid the problematic scales, but efficiency could suffer.

2. We need to better understand how to modify the GAN fro image data to pressure data. The GAN is optimized for image data. Image data consists of values between 0 to 255, making normalization a simple choice. Image data is also padded with zeros around the image in order to simplify the convolution, but it is unclear how much this effects the physically motivated pressure data.

3. Additionally, we need to develop intuition for efficient choices of hyper parameters. For example, within the convolution layer blocks shown in Fig. 10.8, there is batch normalization. Normalizing to a mean of zero and variance of unity makes sense of image data. To train the GAN on pressure data, we need to consider if this choice negatively affects the GAN predictions, since the pressure data is not bounded.

4. Finally, even with a well-trained GAN, integrating the super resolution model into the multigrid method could present challenges.

Francisco Holguin *is a rising 5th year PhD student in the Department of Astronomy at the University of Michigan-Ann Arbor.*

Alex Somers *received his BS in nuclear engineering from North Carolina State University in 2015. From 2016 to 2019, he worked at the Savannah River Site Tritium Facilities as a Shift Technical Engineer. In 2019, he began pursuing a PhD in Nuclear Engineering at Penn State. His research consists of modeling high temperature gas flow for nuclear rocket applications.*

Bibliography

D. Birla. Single image super resolution using gans–keras, 2018.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Ward-Farley, S. Ozair, A. Courville, and Y. Bengio. *Adv. Neural. Procs. Sys.*, pages 2672–2680, 2014.

Y. LeCun and G. Hinton. *Nature*, 521:436–444, 2015.

C. Ledig et al. pages 4681–4690, 2017.

J.M. Reisner L.G. Margolin, C.S. Plesko. A finite-scale model for shock structure. *Physica D*, 403 (132308), 2020.

W.J. Rider L.G. Margolin. A rationale for implicit turbulence modeling. *International Journal for Numerical Methods in Fluids*, 39:821–841, 2002.

L.G. Margolin. Finite-scale equations for compressible fluid flow. *Philosophical Transactions of the Royal Society A*, 367:2861–2871, 2009.

U. Trottenberg, Oosterlee C. W., and A. Schuller. Academic Press, 2001.

W. Yang, X. Zhang, W. Wang, and Xue J. H. *IEEE Trans. Multi.*, 21:3106–3121, 2019.

Chapter 11

Verification for High Explosive Simulations

Team Members
Tali Natan and Olivia Martin

Mentors
Kendra Van Buren, Stephen Andrews, Kyle Hickmann

Abstract

The effective performance of a numerical simulation relies not only on its accuracy but on the ability of users to understand and predict its accuracy. For simulations that exhibit asymptotic convergence–meaning that the numerical solution convergences asymptotically as the grid resolution is refined–the analytical solution to the partial differential equations can be extrapolated and used to approximate the numerical error in the simulation. Standard practices only offer for the extrapolation of a scalar value in the output of the simulation, but a recent report by Hemez, Rider, and Van Buren (2019) proposes an alternative method that allows for the extrapolation of an entire vector solution. In this chapter, we apply the methods developed by Hemez et al. to study the performance of flyer plate and cylinder test simulations in FLAG, a Lagrangian hydrodynamics code used at Los Alamos. Particular attention is directed towards the verification and validation of different artificial viscosity models. The extrapolated solutions are compared against experiments whose specifications directly informed the simulations.

11.1 Introduction

Verification and validation are two crucial processes of computational modeling. They form a basis under which numerical solutions can be interpreted and employed to attain knowledge of a physical system. The ASME V&V 10.1-2012 standard structures verification and validation as follows:

Verification is the process of ensuring that a computational model accurately reflects the theory and equations under which it is built. Verification is divided into two subcategories: code verification and calculation verification. Code verification is the process of affirming that a computational model is free of programming errors and utilizes algorithms that accurately solve the discretized equations.

Calculation verification–often referred to as solution verification–is the process of quantifying the numerical error in a simulation due to the discretization–often referred to as the truncation error–by comparing the numerical solution to the true analytical solution of the partial differential equations (PDEs). Validation is the process of determining how accurately the true analytical solution agrees with empirical measurements (ASME, 2012).

Two problems frequently arise during verification and validation processes. First, the exact analytical solution of the PDEs is not always known and needs to be approximated. This can be done for simulations which exhibit asymptotic convergence, where the discretized solution approaches the true analytical solution as the grid becomes successively finer. In these cases, the existence of a constant rate of convergence can be used to estimate the analytical solution and the truncation error. The ASME V&V 10.1-2012 standard recommend that this be achieved through Richardson extrapolation. Richardson extrapolation is performed by evaluating the numerical solution over (at least) three distinct grid sizes and generating a system of equations modeled by:

$$w_{Exact} - w_h = Ah^p \tag{11.1}$$

where w_h is the numerical solution evaluated on a grid size h, w_{Exact} is an approximation of the analytical solution, p is the order of convergence, and A is a constant. The system of equations can be solved to approximate the order of convergence and the analytical solution for some scalar value w in the output of the simulation.

A second source of inaccuracy, inherent in the Richardson extrapolation, is the restriction of numerical and analytical solutions to scalar quantities. For multi-dimensional problems, the convergence of a scalar metric across numerical simulations may not accurately represent the convergence of the entire system. Recognizing this drawback, researchers at Lawrence Livermore, Sandia, and Los Alamos National Laboratories have proposed an extension to Richardson extrapolation, in which factorization techniques–such as the singular value decomposition–can be used to reduce the dimensionality of a data set by projecting it onto a series of "best fit" vectors (Hemez et al., 2019). Studying the convergence of each vector individually yields an approximation of the "exact-but-unknown" analytical solution. Emphasizing that the approximate analytical solutions cannot yield definitive truncation error, but rather a range of truncation uncertainty, Hemez et al. also propose a method to determine the upper and lower bounds of truncation error.

The objective of this report is to apply the methodologies developed by Hemez et al. to study the convergence of high explosive simulations run on Los Alamos's Lagrangian hydrodynamics code, FLAG. These methods will first be established on one-dimensional flyer plate simulations and expanded to perform validation of Barton and MARS artificial viscosity parameters in two-dimensional PBX 9501 cylinder test simulations.

11.2 Singular Value Decomposition (SVD) Verification Method

11.2.1 Overview of SVD

Singular value decomposition (SVD) is a factorization method used to reduce the dimensionality of a large data set while still capturing its most representative features. Performing SVD on an $(m\ x\ n)$ data matrix A yields an $(m\ x\ n)$ matrix U of orthogonal columns—referred to as left singular vectors—an $(n\ x\ n)$ matrix V of orthogonal columns—referred to as right singular vectors—and an $(n\ x\ n)$ diagonal matrix Σ of scalars–referred to as singular values:

$$A = U \Sigma V^T \tag{11.2}$$

$$\begin{bmatrix} y_1 & y_2 & .. & y_n \end{bmatrix} = \begin{bmatrix} \phi_1 & \phi_2 & .. & \phi_n \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_n \end{bmatrix} \begin{bmatrix} \eta_1 \\ \eta_2 \\ \vdots \\ \eta_n \end{bmatrix} \tag{11.3}$$

In order to attribute physical meaning to the SVD, consider that matrix A holds $(m\ x\ n)$ data points in n-dimensional space. The first right singular vector η_1 is a unit vector along the best fit line passing through the $(m\ x\ n)$ data points and the origin. The corresponding singular value σ_1 is the sum of the projections of all data points along η_1. The second right singular vector η_2 is a unit vector along the best fit line perpendicular to η_1, and its singular value σ_2 is the sum of the projections of all data points along η_2. This procedure is recast for all n right singular vectors and singular values, with each successive singular vector perpendicular to its predecessors. A more detailed explanation of the SVD can be found in Blum et al. (2013).

From this brief description, it is apparent that the amount of information contained within a right singular vector decreases as its number—henceforth referred to as its "mode"—increases, and the information content of each mode is quantified by the magnitude of its singular value. It is also worth noting that the coordinates of the right singular vectors correspond to coordinates in the n-dimensional space. As the SVD is expanded to study high explosive simulations, the n-dimensions will represent the n-grid refinement settings.

11.2.2 SVD Verification

Hemez et al. propose a method of extrapolating the exact-but-unknown solution of an asymptotically convergent multi-dimensional problem by studying the convergence of the right singular vector coordinates. Since these right singular vectors are orthogonal, the convergence of each mode can be studied individually.

The first step in this process is to evaluate the simulation at different mesh sizes and generate a matrix with the computed solutions, organized as columns and placed in sequential order. The solutions must be of equal length, which often requires some form of re-sampling. The SVD is performed on the resulting matrix to produce the left singular vectors, singular values, and right singular vectors.

For each mode, k, an error Ansatz model is applied to study the convergence of the right singular vector coordinates:

$$|\eta_k^{Exact} - \eta_k(\Delta x)| = \beta_k \left(\frac{\Delta x}{\Delta x_0} \right)^{p_k} \tag{11.4}$$

A minimum of three mesh refinement studies are required to solve for the rate of convergence p_k, the regression coefficient β_k, and the extrapolated right singular vector coordinate η_k. A scaling factor of Δx_0 is used in Eq. 11.4 to nondimensionalize the exponential component.

By propagating the extrapolated right singular vector coordinates back through the SVD, the exact-but-unknown analytical solution y_{Exact} is constructed:

$$y_{Exact} = \sum_{1 \le k \le n} \sigma_k \cdot \eta_k \cdot \phi_k \tag{11.5}$$

It is observed by Hemez et al. that high-order modes do not always converge. This is taken to indicate that the modes "represent numerical phenomena that pollute the quality of solutions" (Hemez et al., 2019). For this reason, all diverging high-order modes are exempt from summation in Eq. 11.5.

Hemez et al. use the exact-but-unknown solution to evaluate the L^2 norm of the solution error of the numerical simulations. Acknowledging, however, that the extrapolated solution is only an approximation of the true analytical solution, a set of upper and lower bounds of solution error are derived. These bounds are guided by the minimum and maximum convergence rates of the stable modes:

$$y_{Lower} \cdot \left(\frac{\Delta x}{\Delta x_0}\right)^{p_{Max}} \le \left|\left|\, y_{Exact} - y(\Delta x) \,\right|\right|_2 \le y_{Upper} \cdot \left(\frac{\Delta x}{\Delta x_0}\right)^{p_{Min}} \tag{11.6}$$

$$y_{Lower} = \sqrt{\sum_{1 \le k \le n} \sigma_k^2 \cdot \beta_k^2} \tag{11.7}$$

$$y_{Upper} = \sum_{1 \le k \le n} \sigma_k \cdot |\beta_k| \tag{11.8}$$

Eq. 11.9 is used to determine the uncertainty of each numerical right singular vector coordinate $\eta_k(\Delta x)$. Since the SVD is linear, these uncertainties can be propagated back through the decomposition to determine the numerical uncertainty at each point within the full-field vector solution. All 2^n combinations of the uncertainties $\pm\, U_k(\Delta x)\, \eta_k(\Delta x)$ must be considered in order to determine the full-field uncertainty bounds.

$$\left|\frac{\eta_k - \eta_k(\Delta x)}{\eta_k(\Delta x)}\right| = U_k(\Delta x) \tag{11.9}$$

It is observed by Hemez et al. that accounting for non-converging modes can significantly deteriorate the full-field uncertainty bounds, while neglecting these modes slightly underestimates the bounds.

In this report, we do not seek to prove the validity of the methods developed by Hemez et al. but rather to use these tools for verification and validation of high explosive simulations. Attention to the development of the approach should be directed to Hemez et al. (2019).

11.3 Flyer Plate Verification and Validation

This section serves to formalize and demonstrate the use of SVD for verification of the Shot 1161 PBX 9501 flyer plate simulation. This computational model was constructed to emulate the Shot

1161 experiment performed by Gustavsen, Sheffield, Alcon, and Hill (n.d.). In this experiment, a projectile was launched from a single-stage gas gun towards a sample of the PBX 9501. Upon impact, the projectile generated a planar shock wave in the PBX which initiated detonation. Electromagnetic particle velocity gauges were used to track the velocities of particles within the explosive material. See Section 11.A.1 for more information about the Shot 1161 flyer plate experiment.

In the Shot 1161 FLAG simulation, tracer particles were initialized at equivalent positions to the experimental gauges in order to track and compare point velocities. Simulations in FLAG assume that the flyer plate experiment is perfectly symmetric and model the experiment in one dimension. This simplification yields relatively low computational costs and run times.

11.3.1 Mesh Refinement Study

Verification of the Shot 1161 flyer plate simulation was performed using four mesh sizes: 8.93μm, 12.5μm, 17.86μm, 25μm. This simulation was run using the barton3 artificial viscosity parameters, outlined in Table 11.3, and the AWSD reactive burn model. The second tracer particle was selected for analysis. Fig. 11.1 illustrates the effect of the mesh refinement at the shock front.

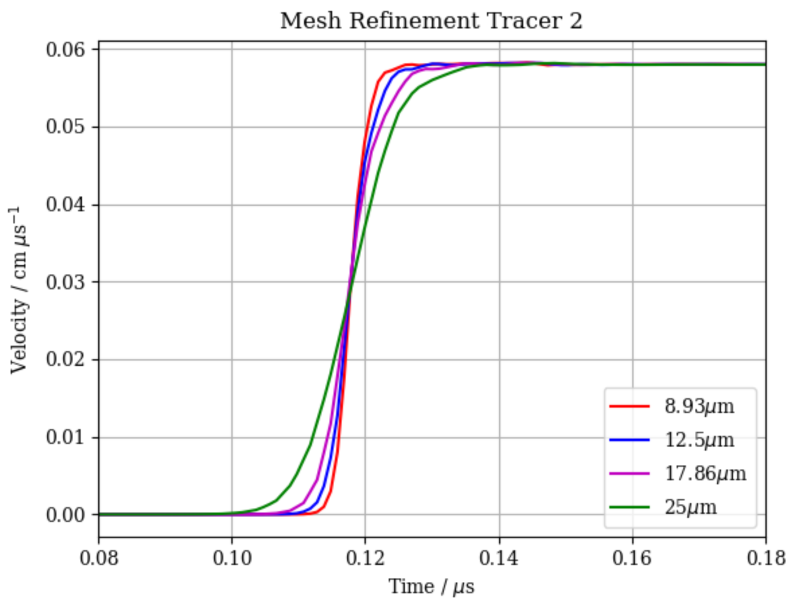


Figure 11.1: **The velocity of tracer particle 2 exhibited greatest dependence on mesh size at the shock front.**

11.3.2 Verification and Validation: Tracer Particle 2

The time step for each simulation was calculated using the Courant-Friedrichs-Lewy (CFL) condition, which defines an upper bound for the time step Δt based on the velocity u and the length interval Δx:

$$\frac{u\,\Delta t}{\Delta x} \le C_{Max} \tag{11.10}$$

In the flyer plate simulations, reducing the the mesh size (Δx) also reduced the time step and increased the length of the output velocity vector. In order to combine the results of the mesh refinement into a matrix for the SVD, the vectors needed to be of equal length. A SciPy interpolation method in Python was used to re-sample the velocity vectors along the time vector corresponding to the simulation with the finest mesh.

An SVD method in Python was used to generate the left singular vectors, right singular vectors, and singular values. The right singular vector coordinates for each mode are presented in Fig. 11.2.

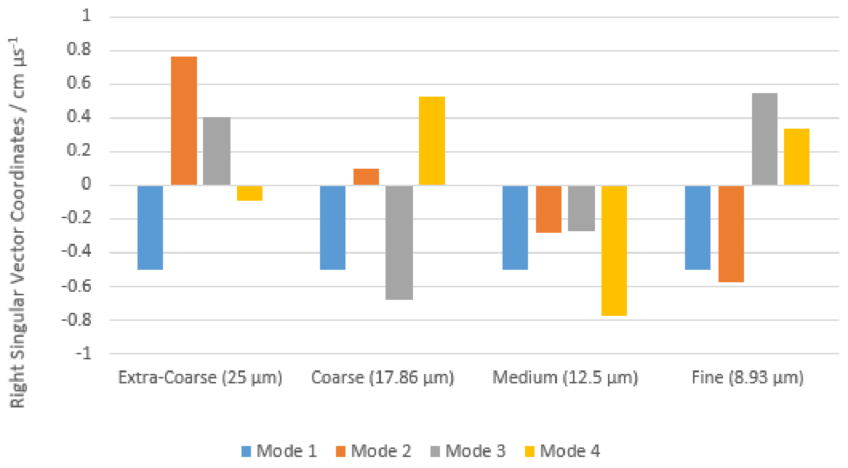


Figure 11.2: **The convergence of each mode is assessed by studying the convergence of the right singular vector coordinates.**

A system of four equations was generated using the error Ansatz model in Eq. 11.4. A value of 25μm was used for Δx_0. A global optimization method in Python was used to solve for the rate of convergence, regression coefficient, and extrapolated right singular vector coordinate by minimizing the L^2 norm of the fitting residuals. This process was completed for each mode. The results are presented in Table 11.1.

Mode, k	**Convergence Rate,** p_k	**Extrapolated Coordinate,** η_k (cm μs^{-1})	**Regression Coefficient,** β_k (cm μs^{-1})	**Singular Value,** σ_k (cm μs^{-1})
1	0.2486	-0.4967	0.0037	4.013
2	2.428	-0.4832	1.250	0.0312
3	-0.001	0.00005	0.4755	0.0053
4	-0.498	-0.2349	0.3850	0.0016

Table 11.1: **Shot 1161 Flyer Plate SVD Results**

The exact-but-unknown solution was obtained by propagating the extrapolated right singular vector coordinates through the decomposition (Eq. 11.5). The third and fourth modes did not converge and were not included in the extrapolated solution. In Fig. 11.3, the extrapolated solution is plotted with the numerical solutions generated in the mesh refinement study.

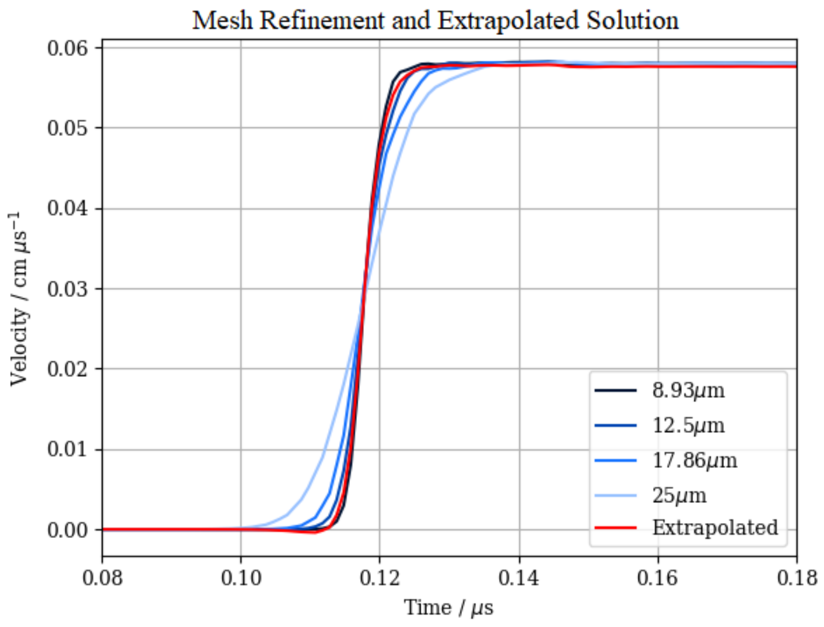


Figure 11.3: **The extrapolated solution was generated using the first two modes. The color gradient indicates the refinement of the mesh.**

A set of upper and lower bounds of the L^2 norm of the solution error were generated using Eq. 11.6, 11.7, and 11.8. These bounds are plotted in Figure 11.4, along with the error evaluated between the extrapolated and numerical solutions.

For a given mesh size, the uncertainty of each right singular vector coordinate was calculated using Eq. 11.9. These uncertainties were propagated back through the decomposition to determine the numerical uncertainty at each point within the full-field vector solution. This process was completed two times: using only the stable modes (1 and 2) and using all four modes. In Fig. 11.5, the full-field numerical uncertainty is plotted for each mesh refinement simulation.

Validation of the Shot 1161 flyer plate simulation was performed by comparing the extrapolated solution to experimental data. This is illustrated in Fig. 11.6. The simulation exhibited very strong performance, particularly considering that this is a multi-dimensional problem being modeled in only one dimension.

11.3.3 Verification and Validation: Tracer Particle 4

During the verification and validation of the flyer plate simulation, we found that the SVD extrapolated solution was not always well-behaved and exhibited a significant overshoot at the shock front. These results were observed when looking at different mesh sizes and when looking at different tracer particles. In this section, we present an example of a less well-behaved extrapolated solution, which was observed while studying the fourth tracer particle.

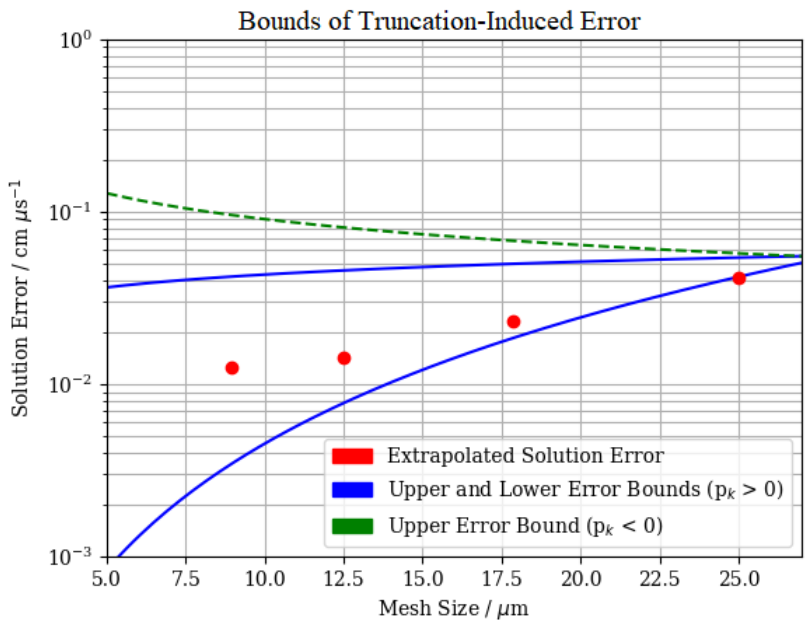


Figure 11.4: **The blue lines illustrate the range of truncation-induced error generated from the first two modes. The dashed green line is the upper error bound generated using the rate of convergence of the unstable mode. The red dots indicate the error between the extrapolated and numerical solutions.**

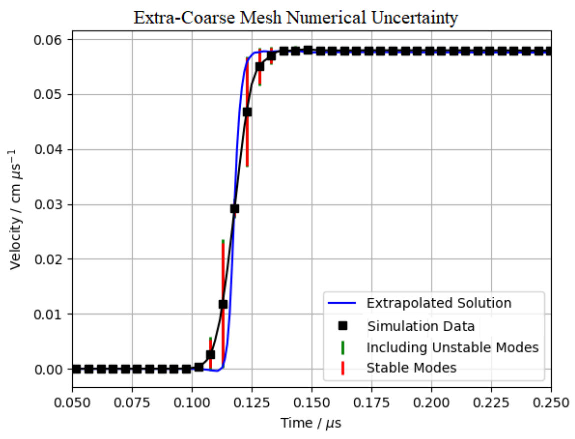
The methods presented in Section 11.3.2 were used to generate the extrapolated solution for the fourth tracer particle. This solution is plotted in Fig. 11.3 along with the results of the mesh refinement study. The convergence rates, extrapolated right singular vector coordinates, and regression coefficients for the four modes are presented in Table 11.2. The mesh sizes used to study the second tracer particle (8.93μm, 12.5μm 17.86μm, 25μm) were also used in this analysis.

As seen in Fig. 11.7, the extrapolated solution significantly overshoots the shock front. It is possible that the SVD is detecting and amplifying the development of instabilities within the fine mesh simulations.

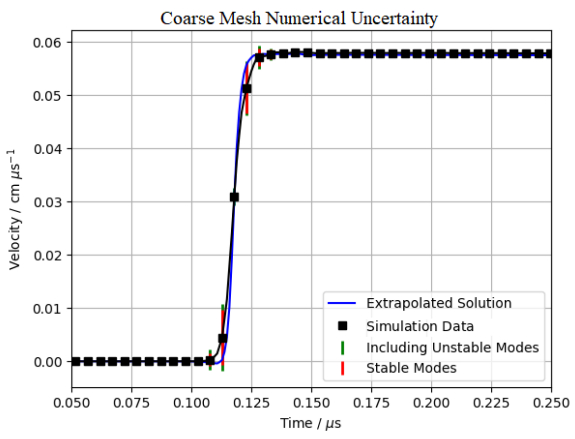
Mode, k	**Convergence Rate,** p_k	**Extrapolated Coordinate,** η_k (cm μs^{-1})	**Regression Coefficient,** β_k (cm μs^{-1})	**Singular Value,** σ_k (cm μs^{-1})
1	2.5051	-0.5002	0.0006	3.8563
2	0.7625	1.7223	2.4440	0.0397
3	-0.1700	-0.0008	0.4541	0.0060
4	-0.0863	0.0064	0.4188	0.0031

Table 11.2: **Shot 1161 Flyer Plate SVD Results: Tracer Particle 4**

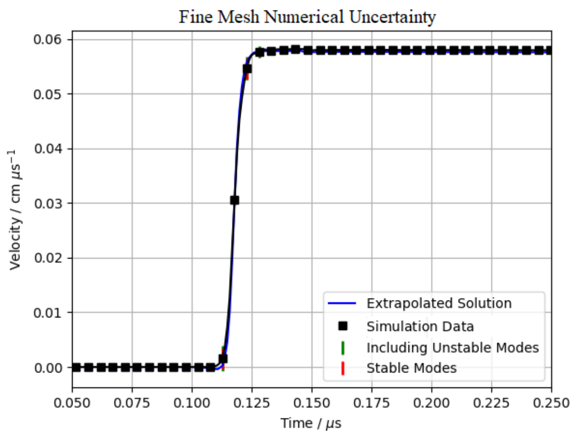
In Fig. 11.8, the L^2 norm of the solution error between the extrapolated and numerical solutions



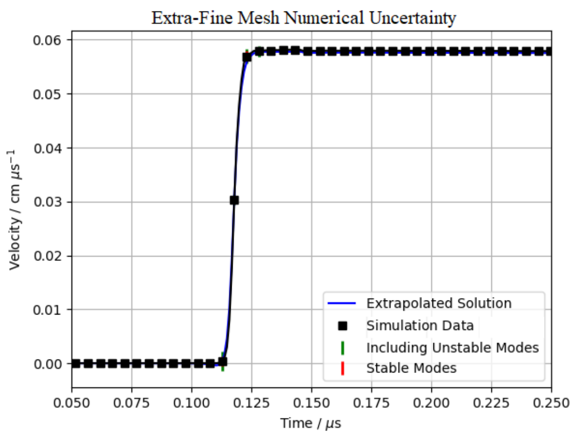
(a) 25μm mesh



(b) 17.86μm mesh



(c) 12.5μm mesh



(d) 8.93μm mesh

Figure 11.5: **The red lines indicate the uncertainty bounds accounting for only the stable modes. The green uncertainty bounds account for all modes–stable and unstable. The extrapolated solution consistently falls within the full-field uncertainty bounds. The bounds do not significantly increase with the addition of the unstable modes.**

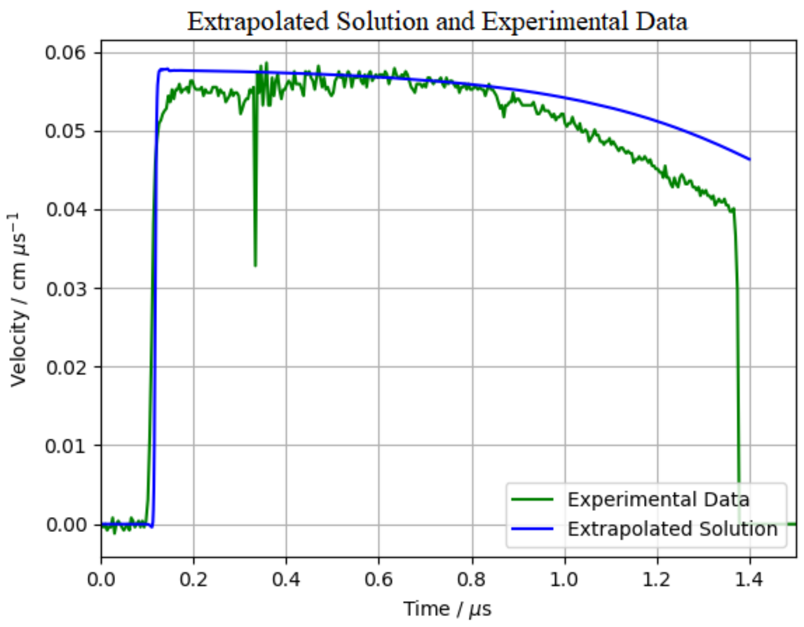


Figure 11.6: **The extrapolated solution demonstrates strong performance when compared against experimental results. There is noise apparent within the experimental data.**

is plotted with the upper and lower uncertainty bounds.

The truncation-induced uncertainty was calculated at each point within the the 25μm mesh simulation. These uncertainties are plotted in Fig. 11.9 along with the extrapolated solution. The uncertainty bounds at the shock front are significantly larger in this example than they were for the second tracer particle.

In cases that exhibited overshoot at the shock front, it is possible that the extrapolations were magnifying small instabilities in the fine mesh simulations. More work, however, must be done to draw an assured conclusion. One path forward may be to dampen the instabilities at the top of the shock by increasing the artificial viscosity settings and analyze how this effects the extrapolated solutions. In Section 11.4.1, we study the effects of lowering the flyer plate artificial viscosity and see significant amounts of ringing at the top of the shock.

11.4 Artificial Viscosity Verification and Validation

In order to resolve shocks, artificial dissipative terms are added to the energy conservation equations to smear out the numerical discontinuities so they can be observed at a fine enough mesh. The technique originally proposed by Von Neumann and Richtmyer (1950) is the basis for artificial viscosity (AV) models. The total artificial viscosity of the system is given by:

$$q = \begin{cases} q_1\rho(\Delta u)a + q_2\rho(\Delta u)^2 & \text{for compression} \\ q_{1n}\rho(\Delta u)a + q_{2n}\rho(\Delta u)^2 & \text{for expansion} \end{cases} \tag{11.11}$$

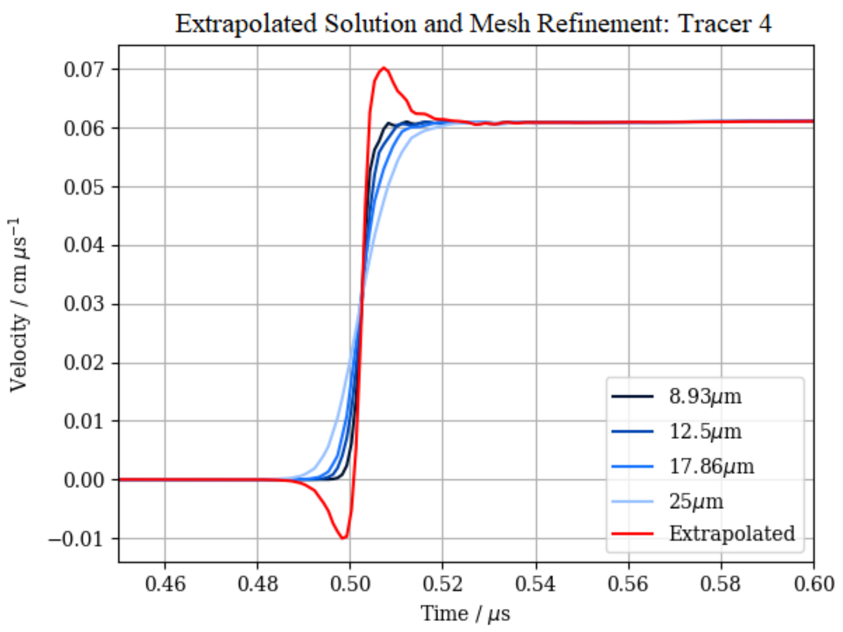


Figure 11.7: **Instabilities appear to develop at the top of the shock and increase in magnitude with the refinement of the mesh.**

where a is the speed sound and Δu is the change in velocity across a cell.

We employed two different artificial viscosity models to modify FLAG: Barton and MARS (Multidimensional Approximate Riemann Solver). The Barton model calculates the viscosity force at the zone edge while the MARS model calculates the viscosity from an approximate Riemann solver based on the linear $U - u$ relationship:

$$U = C_0 + su \tag{11.12}$$

where U is the shock velocity, C_0 is the intercept, and s is the slope. In FLAG, there is an assumed shock impedance of:

$$\mu = \rho U = \rho(aq_1 + q_2\Delta u) \tag{11.13}$$

This means $C_0 \equiv aq_1$ and $s \equiv q_2$.

We examined a total of seven different different AV parameter sets shown in Table 11.3. All sets except for mars4 had been previously examined by Price (2020); investigation of mars4 was suggested by Nathaniel Morgan.

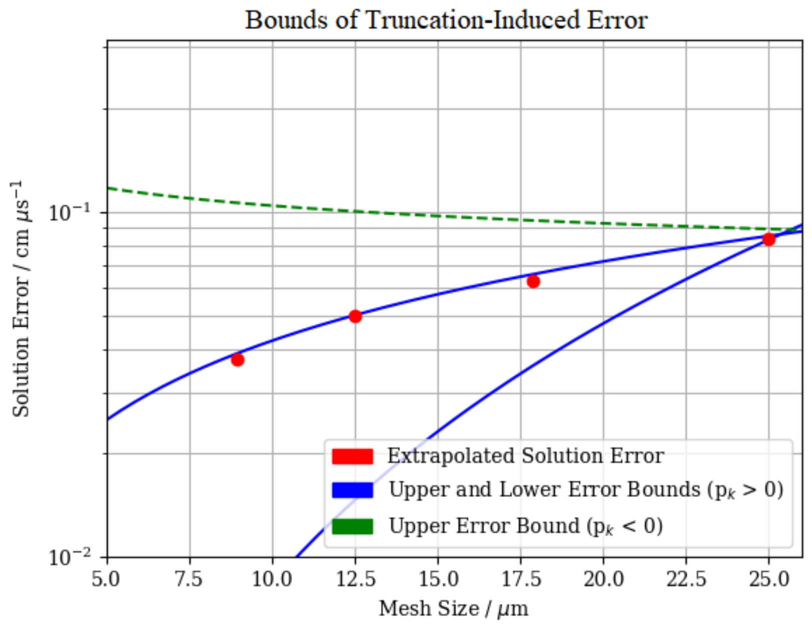


Figure 11.8: **The error between the extrapolated and numerical solutions consistently falls within the blue bounds. The 25μm data point falls to the left of the intersection of the bounds.**

11.4.1 Flyer Plate Shot 1161

The flyer plate Shot 1161 simulation was run using the three different Barton models–MARS is incompatible with one dimensional simulations. In Fig. 11.10, we can see that barton1–the least aggressive AV model–produced the most significant amounts of ringing. Barton3 was able to smooth out the majority of the ringing.

11.4.2 Cylinder Test K12 17232

For the cylinder test simulations, all of the artificial viscosity sets in Table 11.3 were examined. An additional test called mars2.2 was preformed; it has the same parameters as mars2 but with non-default mqtts terms that are shown in Table 11.4. In mars2.2, changing the mqtts parameters relaxed the stiffness of the mesh.

Mesh Refinement Results

For each artificial viscosity setting, the K12 17232 cylinder test simulation was run on four mesh sizes: 25μm, 50μm, 100μm, 200μm. This simulation used the AWSD reactive burn model for the maincharge and the LUND programmed burn model for the booster. The results of the mesh refinement studies are plotted in Figs. 11.11 and 11.12.

The simulations run with barton2, barton3, mars2, mars2.2, and mars3 exhibited strong convergence behavior compared to barton1, mars1, and mars4. These results indicate a potential relationship between the strength of the AV settings and the resulting convergence behavior.

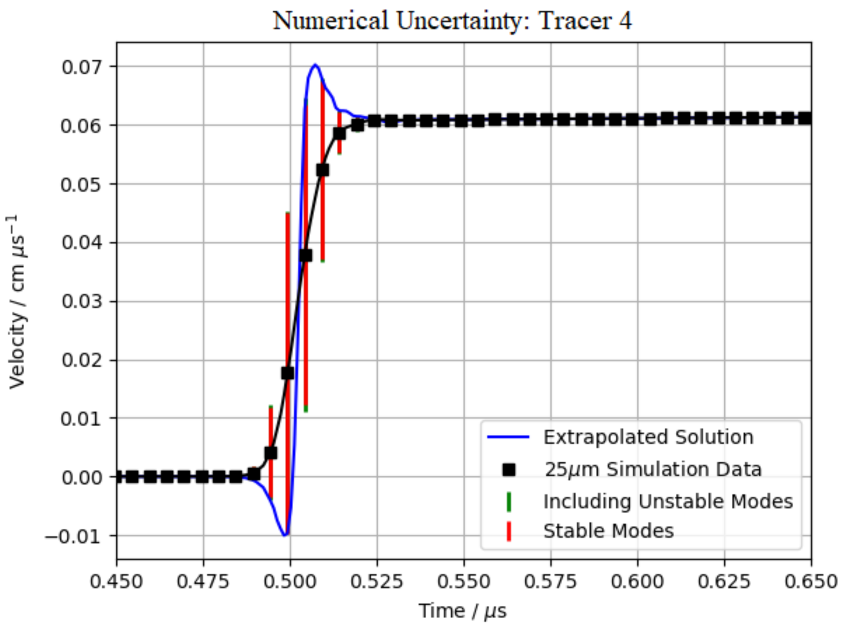


Figure 11.9: **The red uncertainty bounds were generated using only the converging modes. The green uncertainty bounds were generated using all four modes. Significant amounts of numerical uncertainty are exhibited at the shock front.**

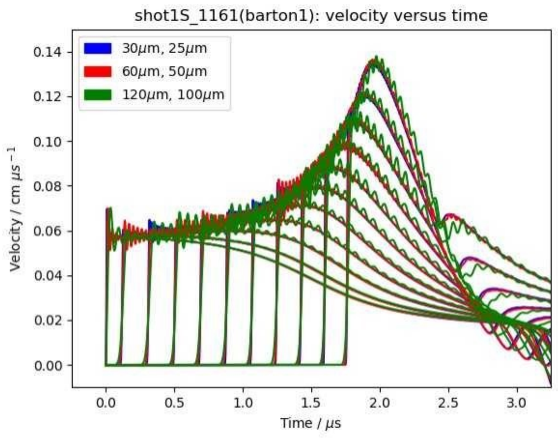
The simulations run with barton1, mars1, and mars4–the three lowest AV models–saw a dramatic increase in detonation velocity with the coarsening of the mesh (the velocity curves shifted to the left) that did not reflect asymptotically convergent behavior. In Section 11.4.3, we study the change in arrival time of the first shock for these AV models more closely. The mesh refinement results for barton1, mars1, and mars4 are plotted individually in Figs. 11.13a, 11.13b, and 11.13c.

SVD Verification and Validation

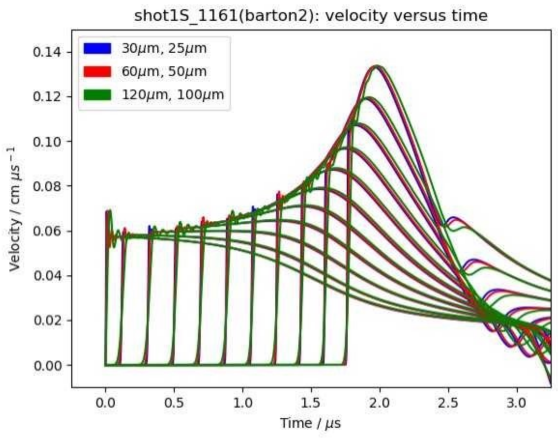
An SVD analysis was completed for each artificial viscosity model. We found that the mode 1 right singular vector coordinates for barton1, mars1, and mars4 did not converge. This is not entirely unexpected, considering the non-convergent changes in detonation velocity observed during the mesh refinement. Since mode 1 contains the largest amount of information about the system, this result indicates that the barton1, mars1, and mars4 simulations were not within the regime of asymptotic convergence. Additional work must be dedicated towards studying these artificial viscosity models in order to extrapolate solutions that accurately approximate the analytical solutions.

The convergence rates of the right singular vector coordinates for the barton2, barton3, mars2, mars2.2), and mars3 simulations can be found in Table 11.5.

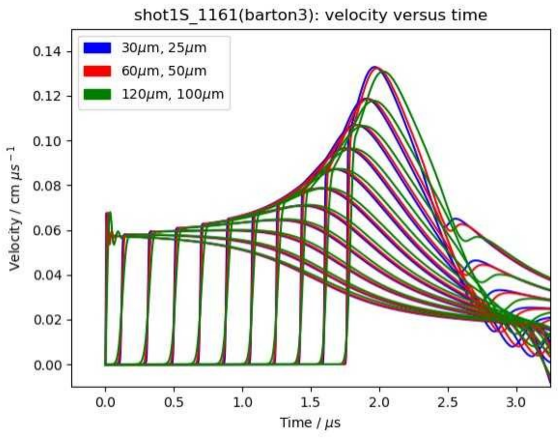
The upper and lower bounds of numerical uncertainty were generated for each artificial viscosity model and plotted with the error between the extrapolated and numerical solutions. These results can be seen in Fig. 11.14.



(a) barton1



(b) barton2



(c) barton3

Figure 11.10: **Flyer Plate Shot 1161 Barton Models. The numbers at the top left indicate the mesh size for the flyer and maincharge respectively.**

Name **Model**	**barton1** **Barton**	**barton2** **Barton**	**barton3** **Barton**	**mars1** **MARS**	**mars2** **MARS**	**mars3** **MARS**	**mars4** **MARS**
q_1	0.0	0.1	0.3	1.0	1.0	2.2	1.0
q_{1n}	0.0	0.1	0.1	0.1	0.5	0.5	0.1
q_2	2.0	2.0	1.33	1.333	1.333	1.333	1.3
q_{2n}	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 11.3: **Artificial Viscosity Models and Parameters**
MARS models use model mars = 2, bjfac = 0.75, model bj = 1 and, bj max = 1

q_1	0.1
q_{1n}	0.1
q_2	1.0

Table 11.4: **mars2.2 mqtts Parameters**

In Fig. 11.14, the error between the extrapolated and fine mesh solution falls outside of the blue bounds for the mars2, mars2.2, and mars3 simulations. There are a few potential reasons why this may have occurred. First, for all three MARS models, the third mode diverged at a relatively significant rate. This may indicate that the fine mesh simulations contained numerical phenomena that were distorting the solutions. These results also raise the question as to whether the simulations were within the region of asymptotic convergence or whether the three mesh refinement simulations were sufficient for performing accurate extrapolation. Future work may be dedicated towards running additional simulations with these AV models and studying the convergence through a collection of four or five mesh refinement studies.

Validation of each AV model was performed by comparing the extrapolated solution to experimental data. This can be seen in Figs 11.15a and 11.15b for the Barton and MARS models, respectively.

In Fig. 11.15, we observe that the extrapolated solutions for barton2, barton3, mars2, mars2.2, and mars3 perform similarly well at capturing the period and amplitude of the velocity oscillations. The Barton models are slightly better at capturing the initial shock. We also observe that the reduction in mesh stiffness (mqtts) had little effect on the mars2 extrapolated solution.

11.4.3 Arrival Time

We wanted to examine our simulations both quantitatively and qualitatively, but finding a value which summarized each simulation proved difficult. After noticing that different mesh sizes within a an AV model exhibited detonation velocities, we decided to calculate the arrival time of each mesh size using a SciPy interpolation method in Python. Our first challenge was defining “arrival time”. Small-scale ringing existed in the data before the first shock began, so we could not choose a cut off velocity value too small lest we identify an irrelevant time and call it the arrival time. Since different mesh sizes also had different rates of acceleration, if we chose a value too large, we might miss the order at which the different mesh sizes absorbed the first shock. We settled on the time at which the velocity exceeds $0.001 cm \mu s^{-1}$. This value was held constant across all of the AV models.

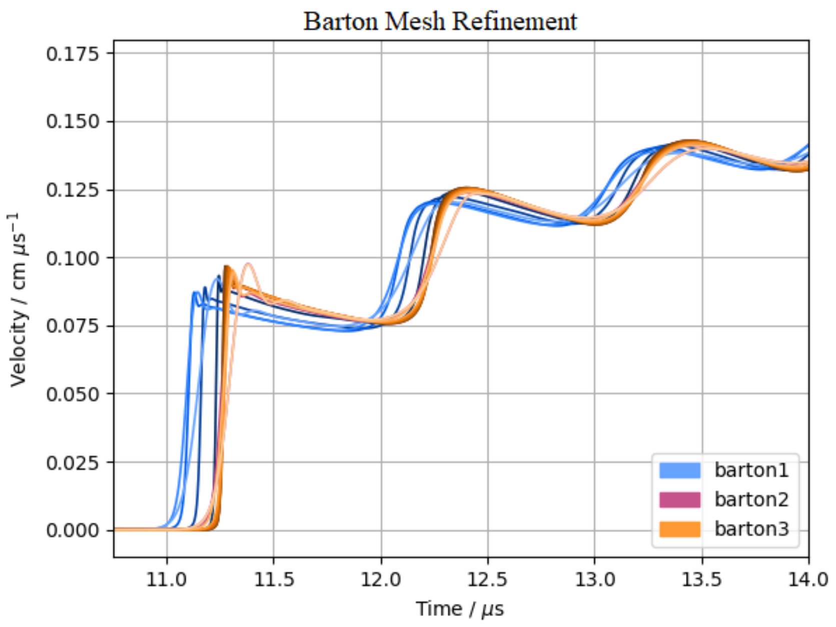


Figure 11.11: **Each color indicates the use of a particular artificial viscosity model. The color gradient illustrates the refinement of the mesh, with the darkest shade corresponding to the 25μm simulation. Barton1 was run on an additional mesh size of 35μm, and barton3 was run on additional mesh sizes of 35μm and 70μm.**

Additional simulations were run for barton1, barton2, and barton3 at mesh sizes of $35\mu m$ and $70\mu m$ to aid this study.

In Fig. 11.16, barton2 and barton3 show a relatively linear relationship between mesh size and arrival time. Assuming R = 1.4, the rate of convergence is about 4.014 and 2.499 respectively; this was calculated using Eq. 11.14 from Hemez et al. (2019).

$$\hat{p} = \frac{log\left(\dfrac{y(\Delta x_M) - y(\Delta x_C)}{y(\Delta x_F) - y(\Delta x_M)}\right)}{log(R)} \tag{11.14}$$

In Eq. 11.14, $y(\Delta x_M)$, $y(\Delta x_C)$, and $y(\Delta x_F)$ are the time values at which the medium, coarse, and fine mesh exceeds $0.001\ cm\ \mu s^{-1}$; R is the mesh refinement ratio, which we approximate as constant.

Barton1, which has the least aggressive AV parameters of the Barton models, shows a relationship between mesh size and arrival time that appears to exponentially grow as the mesh size is refined.

Another way this can be shown is with the visualization application Ensight. Fig. 11.17 shows two cylinder test simulations, both using the barton1 AV model. The color illustrates variation in pressure, measured in megabars. Mesh sizes of 200μm and 25μm are being compared at the same simulation time. The larger mesh is moving faster than the smaller mesh, and the larger mesh is showing patterns that do not exist in the smaller mesh.

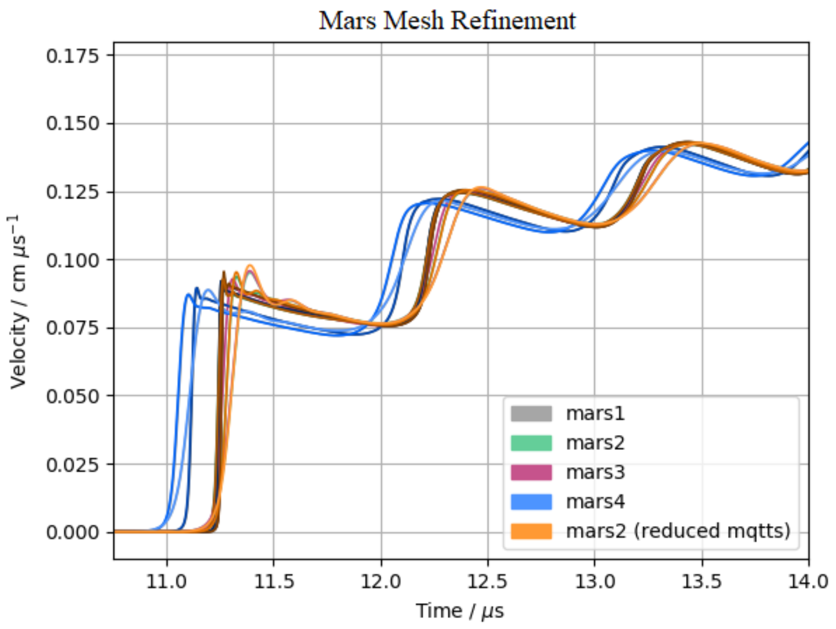


Figure 11.12: **The mesh refinement results from mars1 are located directly beneath mars4. A similar shifting of the shock front is exhibited in the barton1, mars1, and mars4 simulations.**

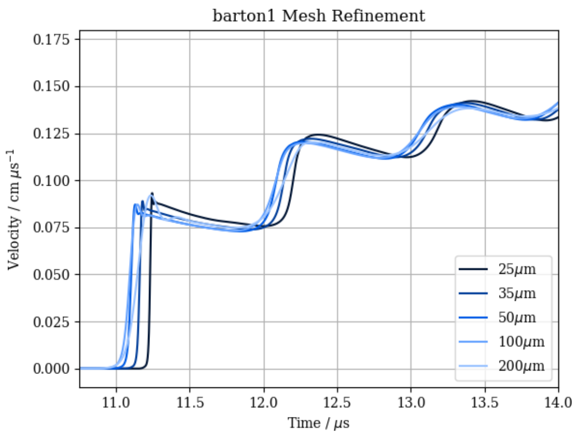
11.5 Conclusion

In this report, the methods developed by Hemez et al. were used to perform verification and validation of flyer plate and cylinder test simulations in FLAG. The SVD was used to decompose the results of each mesh refinement study into a series of “best-fit” vectors, and by studying the convergence of these vectors, an approximation of the full-field analytical solution could be extrapolated. Verification was performed by generating upper and lower bounds of the L^2 norm of the solution error. Validation was performed by comparing the extrapolated solutions to experimental data. As was frequently observed throughout the flyer plate and cylinder test studies, these methods only yield accurate results when in regime of asymptotic convergence.

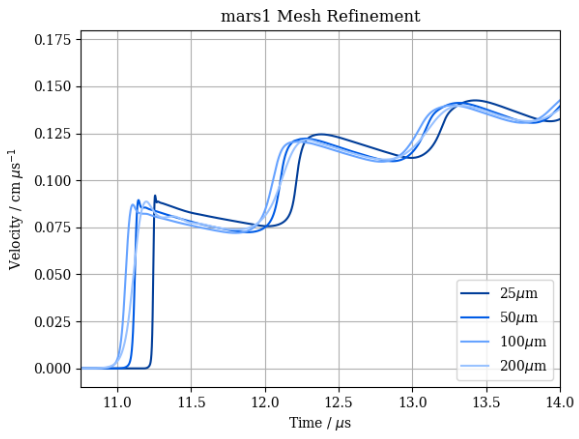
Minimal differences were observed between the Barton and MARS extrapolated solutions for the K12 17232 cylinder test simulations. However, the low artificial viscosity models–barton1, mars1, and mars4–exhibited significantly worse convergence behavior than the more aggressive models. By studying the arrival times of the shock, we found that the detonation velocities of the low AV models diverged with the refinement of the mesh. Overall, the results of this study found that the variations in sizes for a particular AV model were more significant than the variations between different AV models.

11.5.1 Future Work

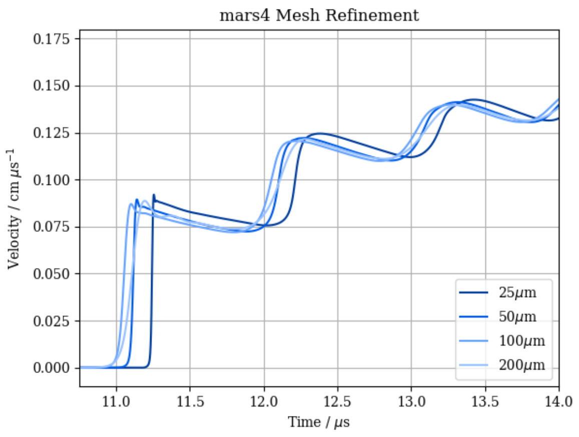
The 25μm resolution simulations frequently crashed between 16 μs and 17 μs due to mesh tangling. Since we only focused on the first few shocks, this was not an issue. If, however, we wanted to



(a) barton1

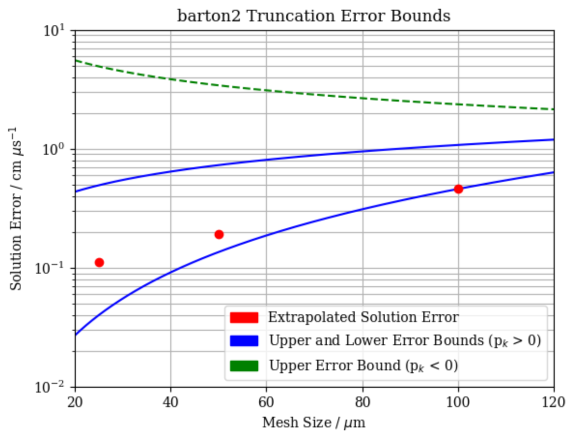


(b) mars1

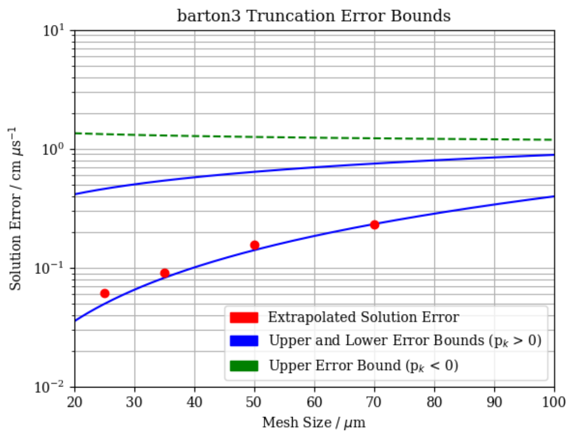


(c) mars4

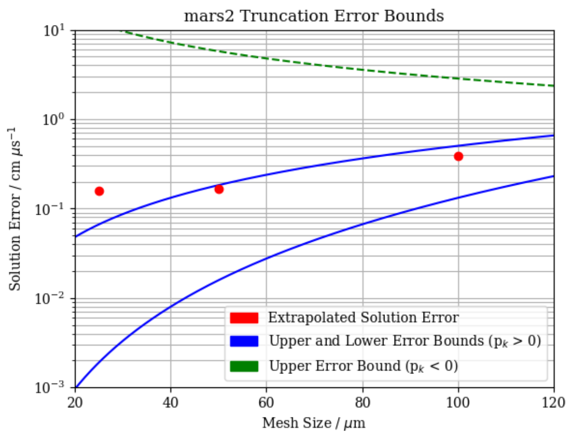
Figure 11.13: **Mesh Refinement of Low Artificial Viscosity Models**



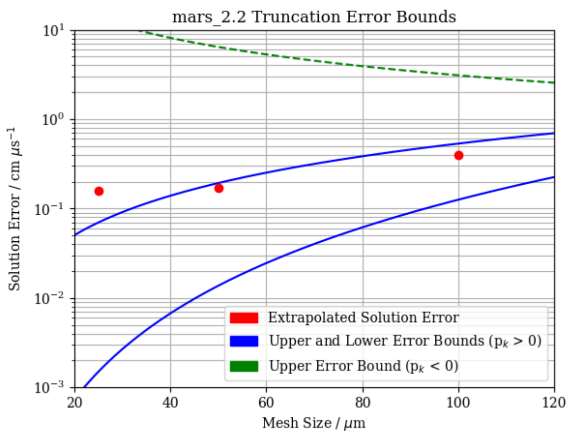
(a) barton2



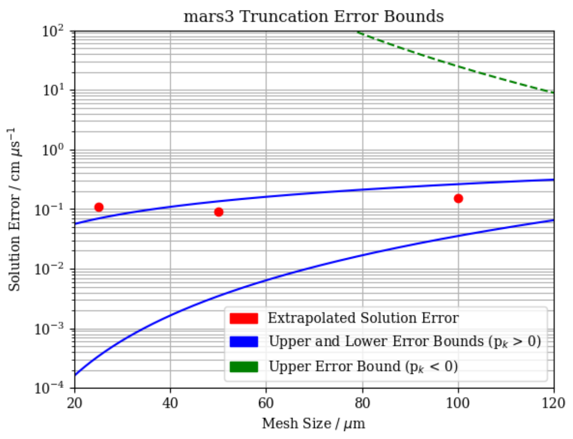
(b) barton3



(c) mars2



(d) mars2.2



(e) mars3

Figure 11.14: **Bounds of Truncation-Induced Uncertainty**

Rate of Convergence					
Mode, k	**barton2**	**barton3**	**mars2**	**mars3**	**mars2.2**
1	0.56	1.35	3.06	3.35	3.19
2	1.76	1.50	1.46	0.96	1.47
3	-0.53	-0.08	-1.02	-5.61	-1.05
4	n/a	0.47	n/a	n/a	n/a

Table 11.5: **Mesh sizes of 25 μm, 50 μm, and 100 μm were used for the mars2, mars2.2, mars3, and barton2 SVD analyses. Mesh sizes of 25 μm, 35 μm, 50 μm, and 70 μm were used for the barton3 SVD analysis.**

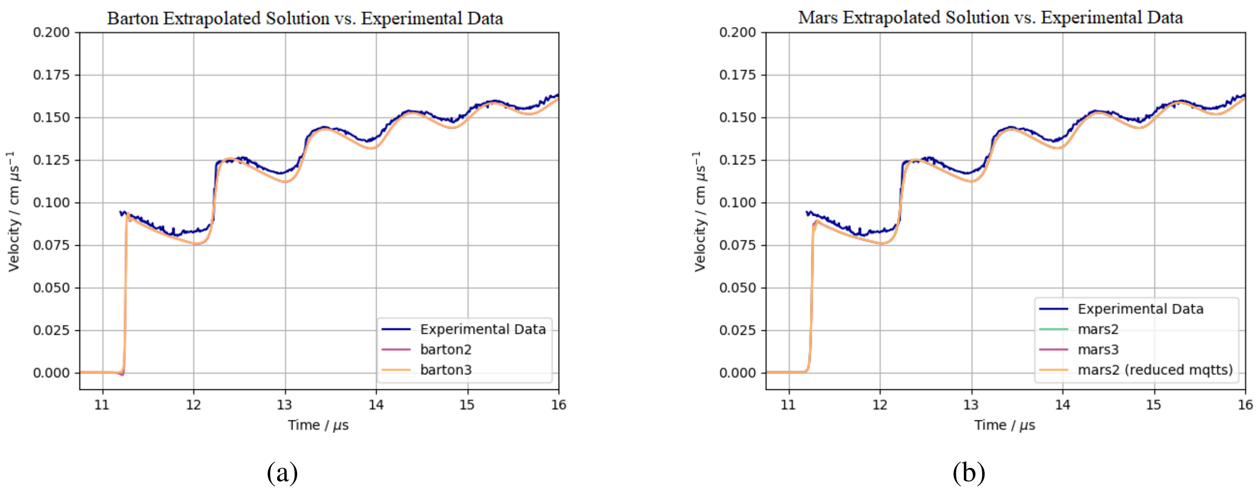


Figure 11.15: **Each solution was time-shifted to intersect the experimental data at a velocity of 0.1 cm μs^{-1}.**

compare the simulations against all available experimental data, we would need to rectify this. One option of dealing with mesh tangling is varying the mqtts parameter–a stiffer mesh may be more resilient against tangling. Another option is to use different ALE settings (which dynamically smooth and remap the simulation onto a better mesh).

Now that we have looked into the Barton and MARS models, it may be worthwhile to see how other AV models compare. Two such models are the VNR model, which is the center solving predecessor to the Barton model, and the BBL model, which is a full tensor AV model using a mimetic approach.

We can also look at applications to more complex geometries that exist within high explosives. In this report, we assumed the PBX 9501 to be homogeneous. It would be interesting to see how FLAG reacts to gaps or other defects in the HE.

Towards the end of the workshop, we started to notice patterns in the arrival time of the first shock in the cylinder test simulations. It is interesting that the shock arrival time appears to have either strong convergence behavior or diverge completely, based on the AV model. We would like to run additional simulations at finer mesh sizes to see if or how this pattern progresses before the

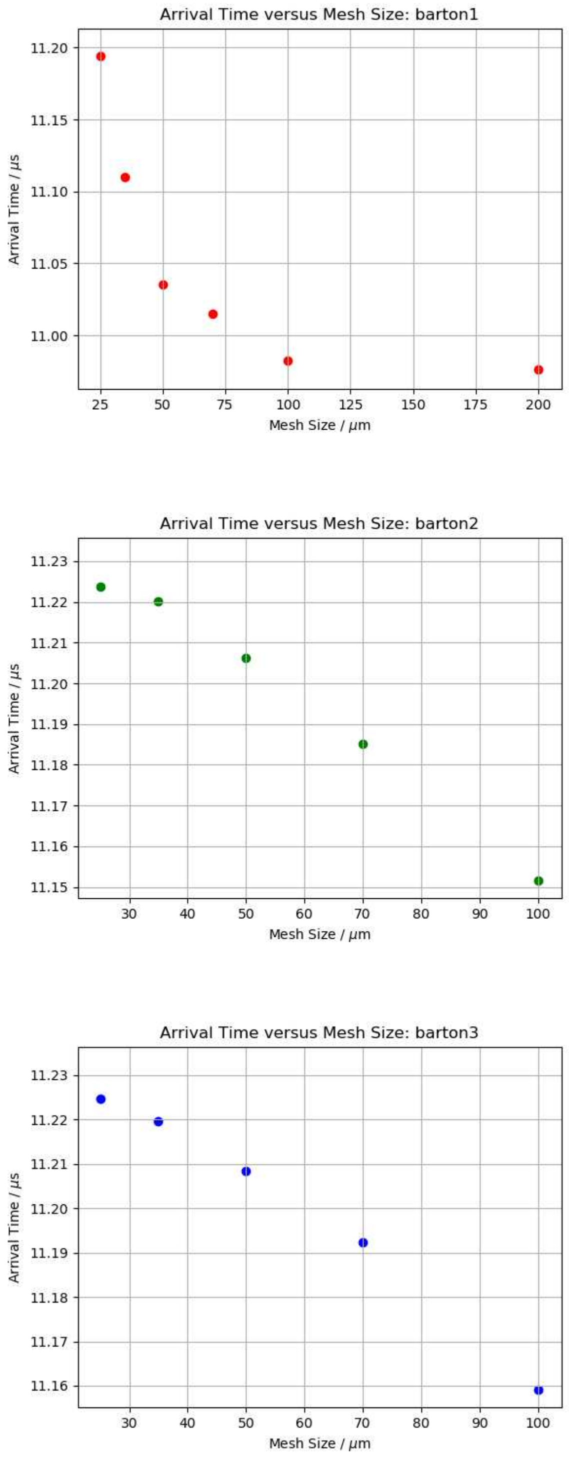


Figure 11.16: **Arrival time versus Mesh Size of barton3**

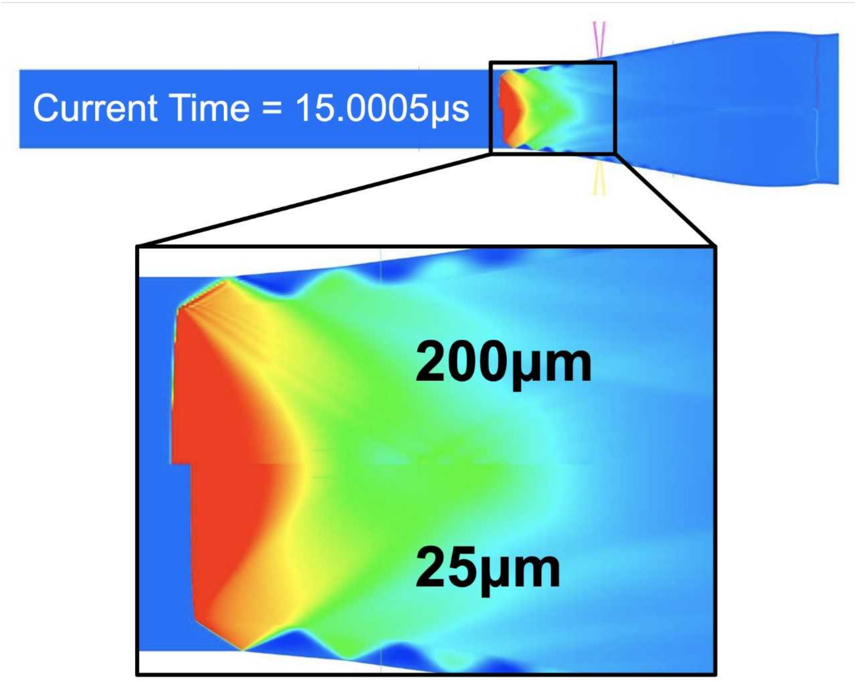


Figure 11.17: **EnSignt plots of barton1 cylinder tests at the same simulation time but different mesh size. Pressure is shown in megabars.**

25 μm mesh. We are also curious to know what the SVD extrapolated solution might look like if we were to adjust all of the mesh refinement simulations for a particular AV model to have the same arrival time.

11.A Experimental Data

Our simulations where based off the criteria of previous flyer plate and cylinder test experiments. In this paper, we compare the experimental data against their corresponding simulations. For the flyer plate, we used Shot 1161 from Gustavsen et al. (n.d.), and for the cylinder test, we used Shot 4 from Pemberton et al. (n.d.). In both the flyer plate and cylinder test shots, the high explosive (HE) PBX 9501 was used. This HE consisted of 95 wt.% HMX, 2.5 wt.% estane and 2.5 wt.% nitroplasticizer.

11.A.1 Flyer Plate Shot 1161

In a flyer plate test, a projectile is faced with a non metallic impactor disk and launched from a single-stage gas gun. A planar shock wave is initiated when the impactor strikes the high explosive (HE) sample. Gauges embedded in the sample at different depths from the impact plane measure the particle velocity and the position of the shock front with time.

In flyer plate Shot 1161, the projectile was shot with a 72-mm bore single-stage gas gun. The impactor was 2.25" (57 mm) in diameter by 0.43" (11 mm) thick, and the explosive sample (or target) was 2" (51 mm) in diameter by about 1 inch (25 mm) thick. The PBX 9501 for the shot was made at the LANL S-site under the supervision of Manny Chavez (ESA-WMM). Material "A" was used in the shot as designated by Gustavsen et al. (n.d.). This sample material was pressed from Holston PBX 9501 molding powder lot 89C730-010 which was manufactured in 1989. Material A refers to pressing number 96-741319 (hydrostatically pressed in a 13.5 x 13.5 x 3.5 inch block) and has a nominal density of 1.826 g/cm^3. Further specifications of Shot 1161 can be found in Table 11.6.

Type	**Initial Density** ρ_0 (g/cm^3)	**Material**	**Impact Velocity** u_I (km/s)	**Tilt** $(mrad)$	**Comment**
A	1.826	z-quartz	0.798	1.5	Tracker broke

Table 11.6: **Specifications of Shot 1161**

The gauge positions for Shot 1161 can be found in in Table 11.7. In this report, we compared gauge one (G1) from the experimental data against tracer two from the simulation.

1st St.	**2nd St.**	**G1**	**G2**	**G3**	**G4**	**G5**	**G6**	**G7**	**G8**	**G9**	**G10**
	0.00	0.44	1.17	1.91	2.64	3.38	4.12	4.85	5.59	6.33	7.07

Table 11.7: **Gauge Positions of Shot 1161 (mm from impact surface)**

There are eleven wave profiles at depths 0 mm to 8 mm into the explosive. The input particle velocity is about 0.06 cm μs^{-1}, but the tracker falls off before the wave reaches a full detonation.

The tracker falling off part-way into this experiment restricted the gauges we could simulate. We chose to look specifically at gauge 1 (G1) because there was no horn at the beginning of the shock and it ended before the tracker fell off.

11.A.2 Cylinder Test Shot 4

In a traditional cylinder test, a metal tube is filled with an explosive and detonated from one end. Photonic Doppler Velocimetry (PDV) is employed to collect data at multiple points along the cylinder at different angles about the circumference. This informs us about the symmetry of the experiment.

Cylinder test Shot 4 was a cylinder test with the HE PBX 9501 from manufacturing lot BAE 03L145-007. The experiment was constructed with copper tubes, which had an inner diameter of $1.0000 \pm 0.0003''$ (0.3 mils tolerance) and an outer diameter of $1.200 \pm 0.001''$ (1 mil). The copper tubes were $12''$ long. The PBX 9501 pellets were machined in $2''$-long sections with an outer diameter of $0.998 \pm 0.001''$. The parts were pressed and machined to specification by WX-8, courtesy of Steve Rivera and company. For more details on the explosive pellets, see Table 11.8. The pellets were lightly lubricated with mineral oil for ease of insertion into the copper tubes.

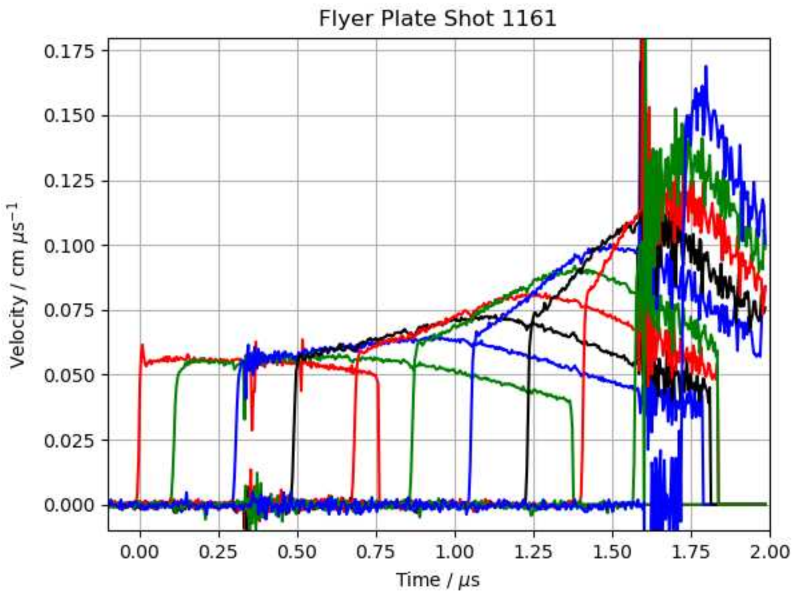


Figure 11.18: **Particle velocity wave profiles from Shot 1161. The burst of noise at approximately 1.6μs is due to the tracker falling off.**

Eight PDV probes were placed at four different distances from the detonator along the cylinder at different angles about the circumference. For information on the probe positions, see Table 11.9

Portion	Identifier	Density
1	18A	1.8356
2	17A	1.8353
3	16A	1.8349
4	15A	1.8343
5	14A	1.8344
6	13A	1.8346

Table 11.8: **Shot 4 Densities by Portion**

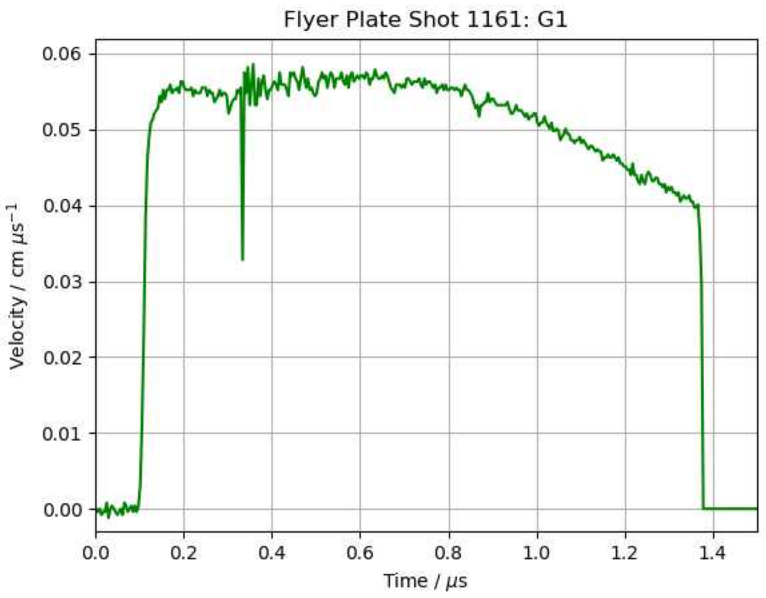


Figure 11.19: **Particle velocity wave profile of G1 from Shot 1161.**

Probe	**Position**	**Face** (mm)	**Spot** (mm)	**Sep** (mm)	**Angle** (deg)
1	1-1	85.0	76.0	74.000	7.028
2	1-2	111.0	102.0	74.000	7.028
3	1-3	136.5	130.0	74.000	5.088
4	1-4	128.0	135.0	74.000	5.477
5	2-2	112.0	102.5	75.000	7.316
6	2-4	127.5	136.5	75.000	6.934
7	3-2	111.0	103.0	72.273	6.404
8	3-4	127.5	133.5	71.909	4.837

Table 11.9: **Shot 4 Probe Positions**

Olivia Martin *is a rising senior at Tufts University majoring in Mechanical Engineering. She is interested in fluid dynamics and computational fluid modeling.*

Tali Natan *is a rising senior at Kenyon College where she majors in Physics with a concentration in Scientific Computing. She is interested in nuclear astrophysics, especially contact binary systems.*

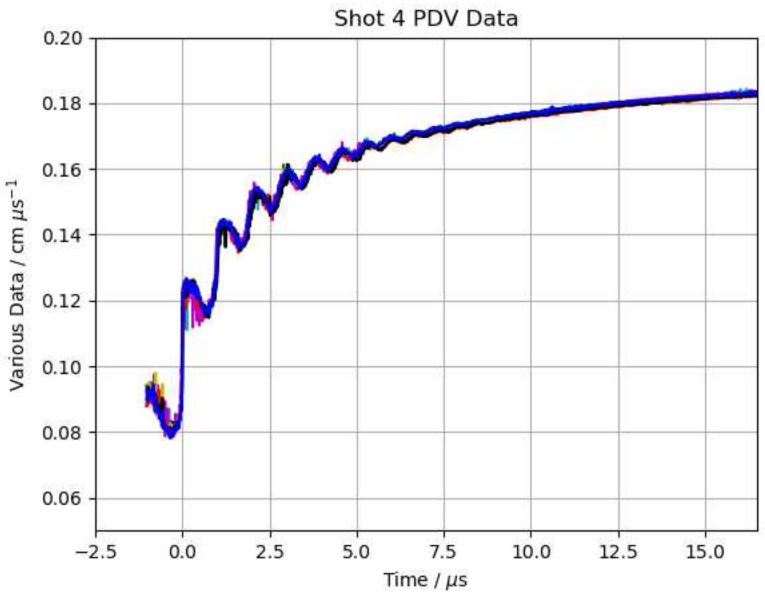


Figure 11.20: **The data provided by the PDV probes showed no dependence on the position along the tube. To help illustrate this, the velocity curves were shifted and plotted on top of each other.**

Bibliography

ASME. *ASME V&V 10.1-2012: An Illustration of the Concepts of Verification and Validation in Computational Solid Mechanics*. American Society of Mechanical Engineers, 2012.

A. Blum, Hopcroft J., and R. Kannan. *Foundations of Data Science*. 2013.

R. L. Gustavsen, S. A. Sheffield, R. R. Alcon, and L. G. Hill. Shock Initiation of New and Aged PBX 9501 Measured with Embedded Electromagnetic Particle Velocity Gauges. Technical Report LA-13634-MS, Los Alamos National Laboratory, Los Alamos, NM, USA, n.d.

F. Hemez, W. Rider, and K. Van Buren. Functional Convergence of Multi-dimensional Numerical Solutions. Technical Report LA-UR-19-28166, Los Alamos National Laboratory, Los Alamos, NM, USA, August 2019.

S. Pemberton, T. Sandoval, T. Herrera, J. Echave, and G. Maskaly. Test Report for Equation of State Measurements of PBX 9501. Technical Report LA-UR-11-04999, Los Alamos National Laboratory, Los Alamos, NM, USA, n.d.

M. A. Price. ZND Verification Tests for Reactive Burn Models in FLAG. Technical report, Los Alamos National Laboratory, 2020.

J. Von Neumann and R. D. Richtmyer. A method for the numerical calculation of hydrodynamic shocks. *Journal of Applied Physics*, 21.

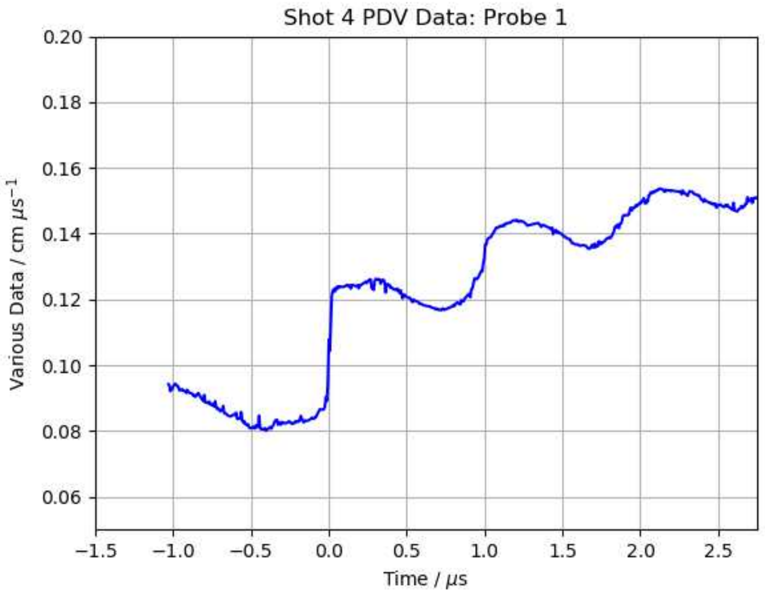


Figure 11.21: **Probe one is plotted above.**

Chapter 12

Photon Transport in Warm Dense Matter

Team Members
William Johns, Jackson White

Mentors
Charles Starrett, Nathaniel Shaffer, Chris Fontes, Nathanael Gill

Abstract

Accurate computation of opacities in plasmas is important in many applications including astrophysics and inertial confinement fusion. An important quantity that underlies the opacity models is the ion distribution, that is the fraction of the plasma consisting of each ionization stage of its atoms. Current methods available at LANL to compute the ion distribution are based on isolated atom approaches that do not take the density effects in the plasma into account in their initial computations and add them later via perturbation. The TARTARUS average atom model developed at LANL takes the density effects of plasma into account from the beginning. We have extended the capabilities of the TARTARUS model to compute the ion distribution in a plasma. We believe this can be used to improve computation of opacities with TARTARUS Shaffer et al. (2017) especially when the density effects are strong.

12.1 Project Overview

TARTARUS is an average atom code used to compute the parameters of an average atom in a plasma, from Starrett (2015). An average atom is an approximation where in place of integer electron occupations average fractional occupations from Fermi-Dirac statistics are used to approximate the thermal distribution of electron configurations of one averaged atom in a plasma.

12.1.1 Tartarus Motivation

There are other codes currently available which attempt to similarly calculate ionization stage occupation populations, notably ChemEOS from Hakel and Kilcrease (2004), which we use throughout our project as a comparison to our results. ChemEOS uses an isolated atom approach, while TARTARUS uses a Self-Consistent Field (SCF) approach, to include plasma effects. This is an iterative approach, which first assumes an initial effective potential V^{eff}, then uses that potential to calculate electron orbitals by solving the Schrödinger equation, and then uses the resultant electron densities to solve for a new V^{eff}, iterating until the two effective potentials agree to some specified difference value.

Our expectation is that while an isolated atom approach like ChemEOS should be more accurate for the case of a relatively low density plasma, where atomic configurations are more like isolated configurations, TARTARUS should be more accurate for relatively high density plasma, where the electron orbitals and energies vary more significantly from those of an isolated atom.

12.1.2 Tartarus Modifications

To accomplish this project we made two primary modifications to TARTARUS. First we replaced the average fractional occupations with integer occupations from an input file. TARTARUS then models the behavior of a specific electron configuration in the plasma. The second is we replace the local density approximation (LDA) bound electron interaction energy with the more accurate correlation corrected Hartree-Fock interaction energy. We then use the configuration energies and free electron densities of these configurations to compute the ion distributions.

Neither of these modifications are run by default. Tartarus preforms an average atom calculation unless otherwise specified in the input file, and the energy corrections are not automatically added to any of the Tartarus output variables. The energy correction values are instead printed to the output file.

12.1.3 Ion Distribution Calculations

We implemented two different methods for computing the ion distributions. The first is the well established Saha-Boltzmann approach, which uses the Saha equation to calculate the ratios between successive ionization stages, and the Boltzmann equation to calculate configuration populations within an ionization stage, under the condition of local thermodynamic equilibrium (LTE). The second is a variational methodology based on a constrained minimization of the free energy of the system. We compare these methods with each other, ChemEOS, and other results from the literature.

12.2 Integer States in Tartarus

To replace the Fermi-Dirac occupation statistics with integer occupation numbers we decided to input the occupations from a file and read them into TARTARUS. If the read_states variable in the TARTARUS input file is `.true.` then TARTARUS searches for a file called “states” in the directory from which it was called. “states” contains the occupation by orbital in the following format.

# lines
n_1 l_1 e_1
n_2 l_2 e_2
n_3 l_3 e_3
...

Here $\#lines$ is the number of lines in the states file for TARTARUS to read (not including the first line of the file. Each subsequent line contains a subshell n,l and the number of electrons occupying the subshell e separated with spaces. For example for the configuration $1s^2 2p^1 4d^2$ the states file should read.

```
3
1 0 2
2 1 1
4 2 2
```

Each time TARTARUS would use a Fermi-Dirac occupation factor for core states in a calculation it replaced by the appropriate factor computed from the states file

$$fd_k = \frac{e_k}{2(2l_k + 1)} \tag{12.1}$$

After TARTARUS converges, a final check is performed to test if all of the states in the "states" file were found to be bound. If an electron was input in a subshell that was found not to be bound then the output message, "The bound state n=5 l=2 in states input file was not found ", will be written to output where n, l will indicate the subshell that was not bound, and TARTARUS will terminate.

12.3 Generating the states file

To generate 'states' files for different electron configurations we wrote the script (config_gen2.py). Given an atomic number N and a maximum subshell, it computes all the configurations up to double excitations (2 electrons can be excited from ground state to any subshell considered) for each number of bound electrons up to N. The results are stored in a folder called "double" (for double excitations). Each file is named "ground_aa_bb_cc" where "aa" is the number of bound electrons, "bb" is the index of the first excitation (not physically meaningful) and "cc" is the index of the second excitation (again not physically meaningful). To consider only a subset of possible bound electrons, the "ground_aa*" files should be deleted for each "aa" not be considered.

12.4 Running multiple states interactively

The "batchfig1.py" script is used to run all the states generated by the 'config_gen2.py' script in an interactive session. It must be run in a folder containing a copy of the TARTARUS executable, a TARTARUS "input" file with `read_states=.true.`, and the folder "double" containing the configurations to be run. For each file ground_aa_bb_cc in "double" a directory "results_aa_bb_cc" is created. The file "ground_aa_bb_cc" is copied into this folder and renamed "states", a copy of the TARTARUS executable and the input file are then copied to the directory. TARTARUS is then executed in this directory and the output is sent to "results/TARTARUS_aa_bb_cc.out". Upon completion, the "results_aa_bb_cc" directory is deleted.

12.5 Sorting the results

The config_sorter.py script is used to sort all the results from running batchfig1.py. It creates the directories, "goodstates", "notconverged", "good_configs", and "bad_configs". Each ".out" file is then searched for the strings 'did not converge' and 'The bound state' indicating that TARTARUS did not converge or that the "states" file contained electrons in a subshell that was not found be bound. If either string is found, the corresponding "ground_aa_bb_cc" is moved from "double" to "bad_configs" and the "TARTARUS_aa_bb_cc.out" file is moved to "notconverged". If neither string is found, the file is checked for the final output lines 'Ionization Level =' or 'Run Complete'. If either one of these is found, the corresponding files are moved to "good_configs" and "good_states" respectively. If none of these lines are found, then the TARTARUS run was not completed. No files are moved, which allows the batch to be continued by running batchfig1.py again, as the unfinished states are still in the "double" directory.

One of the most common reasons for a run not to complete is that the KSM routine will fail to find a chemical potential (mu), resulting in the error message, ":STOP ksm_mu_finder: Failed to find mu". This can either be caused because the configuration in question does not exist, or it can be caused by the selected 'lmax' parameter being too large, in which case NaN values will show up in the relevant portion of the output file. For this reason, runs that both do not complete due to this error, and have NaN values in the relevant location of the output file, will not be moved from "double", so that they can be rerun with a lower "lmax".

12.6 Running multiple states as batch

The wbatch.sct script runs the batchfig1.py script and then the config_sorter.py script. Memory can be an issue, but using 3 nodes with 35 processes in batchfig1.py appears to work consistently with the full TARTARUS model.

12.7 Variational Method

Implemented in the "guidance.py" script is the following variational method for finding the ionization probabilities. We compute the ionization probability distribution by minimizing the free energy of the system. The free energy Ω is given by

$$\Omega = \sum_x g_x W_x (F_x + Tln(W_x)) - B(\sum_x W_x - 1) - C(\sum_x W_x n_x^0 - \bar{n}_e^0) \tag{12.2}$$

Where the sums are over the configurations and T=temperature, F_x=free energy, n_x^0=electron density, $\bar{n}_e^0$= average electron density from TARTARUS run as an average atom model, B and C are Lagrange multipliers, g_x is the subshell degeneracy and W_x is the desired probability.

The constraints guarantee that the sum of probabilities is one, and that the average electron density is recovered. Minimizing with respect to W_x we find

$$W_x = g_x \exp(-\frac{1}{T}(F_x + T - B - Cn_x^0)) \tag{12.3}$$

Letting

$$A = \exp(-\frac{1}{T}(T - B)) \tag{12.4}$$

we can rewrite

$$W_x = g_x A \exp(-\frac{1}{T}(F_x - Cn_x^0)) \tag{12.5}$$

To determine A and C we enforce the constraints:

$$A = (\sum_x g_c \exp(-\frac{1}{T}(F_x - Cn_x^0))^{-1} \tag{12.6}$$

and

$$\sum_x \frac{g_x \exp(-\frac{1}{T}(F_x - Cn_x^0))}{\sum_{x'} g_{x'} \exp(-\frac{1}{T}(F_{x'} - Cn_{x'}^0))} n_x^0 = \bar{n}_e^0 \tag{12.7}$$

Which we solve with Newton's method with secants. It should be noted that these function often have long flat tails making the root solving sensitive the initial value for C. We have been ad-hoc changing the initial C value based on the problem. Using the bisection method may be better than Newton's or could be used to initialise a good guess for Newton's method. The values of F_x may be large in which case computation will results in overflow errors. We compute instead

$$\sum_x \frac{g_x \exp(-\frac{1}{T}(F_x - F0 - Cn_x^0))}{\sum_{x'} g_{x'} \exp(-\frac{1}{T}(F_{x'} - F0 - Cn_{x'}^0))} n_x^0 = \bar{n}_e^0 \tag{12.8}$$

where

$$F0 = \min(F_x) \tag{12.9}$$

If there is a value in F_x near zero it may be necessary to replace $\min$ with the mean to prevent the overflow error. On some examples we have been using overflows still occur in the root finding for C, but the built-in Newton's method appears to handle these well.

12.8 Hartree-Fock Energy Correction

The TARTARUS code uses Finite Temperature Density Functional Theory, which does a relatively poor job of modelling the exchange-correlation configuration energy between bound electrons, as in Wilson et al. (1995). In the Local Density Approximation (LDA), the exchange-correlation energy is given as a functional of the local density, so that the exchange-correlation energy between bound electrons is given by

$$E_{xc,bb} \equiv \int d^3r \left\{u_x\left(n_b(r)\right)\right\} \tag{12.10}$$

Where some functional u_x of the bound electron density n_b must be chosen. In addition, the direct LDA interaction energy between bound electrons is given by

$$E_{d,bb} \equiv \int d^3r d^3r' \left\{ \frac{n_b(r)n_b(r')}{|\vec{r} - \vec{r'}|} \right\} \tag{12.11}$$

Following Piron et al (2013), we replace the LDA total interaction energy, between bound electrons, of some electron configuration C with the Hartree-Fock interaction energy

$$\left(E^C_{d,bb} + E^C_{xc,bb}\right) \cong \frac{1}{2} \sum_{\alpha,\beta} q^C_\alpha (q^C_\beta - \delta_{\alpha,\beta}) V_{\alpha\beta} \tag{12.12}$$

Where q^C_α is the electron occupation number of subshell α in configuration C. $V_{\alpha\beta}$ is the shell-shell interaction energy and is given by the expressions:

$$V_{\alpha\beta} \equiv \mathcal{F}^0(\alpha,\beta) - \frac{1}{2} \sum_{k=|\ell_a - \ell_b|}^{\ell_\alpha + \ell_\beta} \begin{pmatrix} \ell_a & k & \ell_b \\ 0 & 0 & 0 \end{pmatrix}^2 \mathcal{G}^k(\alpha,\beta) \text{ if } \alpha \neq \beta \tag{12.13}$$

$$V_{\alpha\beta} \equiv \mathcal{F}^0(\alpha,\beta) - \frac{2\ell_\alpha + 1}{4\ell_\alpha + 1} \sum_{k>0} \begin{pmatrix} \ell_a & k & \ell_a \\ 0 & 0 & 0 \end{pmatrix}^2 \mathcal{F}^k(\alpha,\beta) \text{ if } \alpha = \beta \tag{12.14}$$

$\mathcal{F}^k$ and $\mathcal{G}^k$ are the direct and exchange Slater integrals, respectively, and following Faussurier et al (2018) are given by:

$$\mathcal{F}^k(a,b) = e^2 R^k(\alpha,\beta,\alpha,\beta) \tag{12.15}$$

$$\mathcal{G}^k(a,b) = e^2 R^k(\alpha,\beta,\beta,\alpha) \tag{12.16}$$

$$R^k(a,b,c,d) = \int_0^{+\infty} dr_1 \int_0^{+\infty} dr_2 P_a(r_1)P_b(r_2)P_c(r_1)P_d(r_2) \frac{r^k_<}{r^{k+1}_>} \tag{12.17}$$

Where $P_a(r)$ denotes the radial wave function for some electron a, and the notation $r_<$ and $r_>$ denote $\min(r_1, r_2)$ and $\max(r_1, r_2)$, respectively.

For this project, the Hartree-Fock interaction energy, Direct LDA energy, and Exchange LDA energy are all calculated individually and then printed at the end of the TARTARUS output file. At the moment the HF energy correction, equal to (Hartree-Fock interaction energy - LDA interaction energy), is NOT automatically added to any of TARTARUS's outputs values.

12.9 Correlation Correction

The Hartree-Fock Interaction Energy only partially considers correlation effects between bound electrons. This leads to a configuration energy value that is too high, so an additional correlation energy correction term is needed. Following Cowan (1981), we take the correlation energy correction,

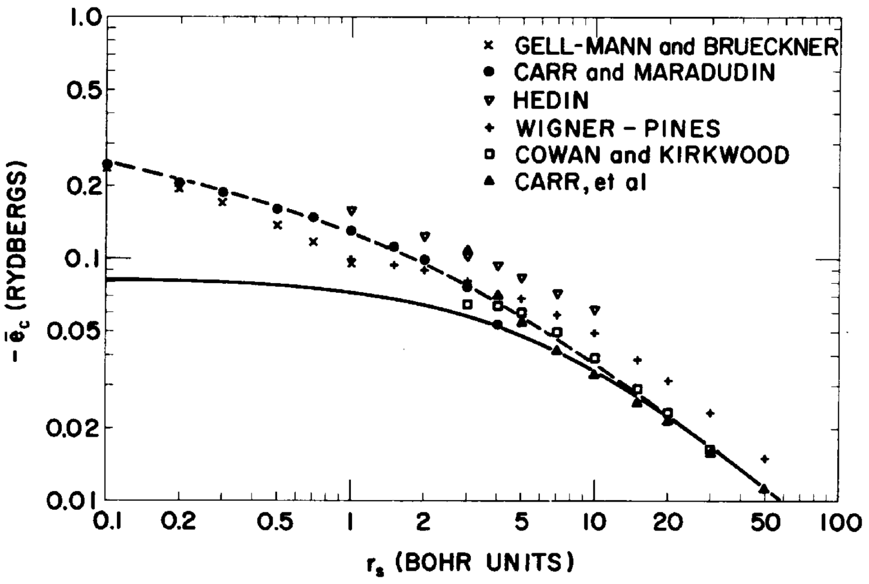


Figure 12.1: e_c, the average correlation energy per electron, as a function of r_s, the radius of the average volume per electron, where the plotted points are theoretical values of e_c in various approximations, the dashed line is a fit to the plotted points, and the solid line is the semi-empirical curve given in equation 12.19, Cowan (1981)

E_C, to be a function of the electron radial wavefuntions, $P(r)$, and a semi-empirical value, e_C, the average correlation energy per electron. Note that the summation below sums over each individual electron i, not over electron subshells.

$$E_c = \sum_{i=2}^{N} \int_0^{\infty} P_i^2(r) e_c^i(r_s) dr \tag{12.18}$$

$$e_c(r_s) = -\left[4(r_s+9)^{1/2} + \frac{3}{4} 1.142\, r_s\right]^{-1} \tag{12.19}$$

Where r_s is taken to be the radius which defines a sphere whose volume is the average volume per bound electron, and is therefore a function of r, as defined in the expression:

$$r_s = \left[\frac{1}{3r^2} \sum_{j=1}^{i-1} P_j^2(r)\right]^{-1/3} \tag{12.20}$$

Shown in figure 12.1, e_c, the solid curve, is an interpolation function found from considering the high-density limit, where $e_c \cong -0.08$ Rydbergs per electron, an empirical observation, and the large r_s limit of the dashed curve in figure 12.1, which is a fit to the plotted theoretical values.

Together, both the correlation correction and the Hartree-Fock correction contribute a configuration energy correction which increases the magnitude of the configuration energy, making it more negative, as shown in figure 12.2

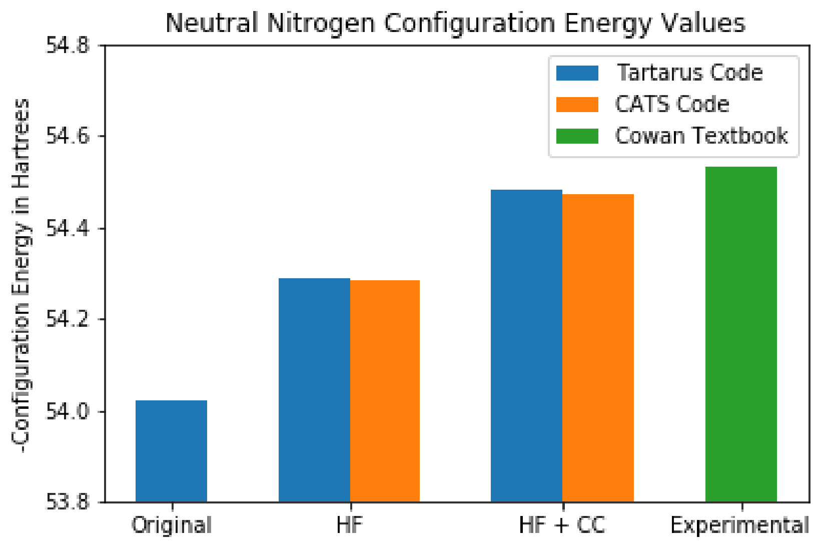


Figure 12.2: Configuration Energy Values for Neutral Nitrogen. 'HF' denotes Hartree-Fock corrected, 'CC' denotes correlation corrected. CATS is a LANL code based on Cowan's atomic structure code, and the experimental value also comes from Cowan's textbook

12.10 Saha-Boltzmann Calculation

In addition to the Variational Method, we also attempted to use the Saha-Boltzmann equations.

The Boltzmann equation gives the ratio of the density of atoms with N electrons in electron configuration j, to the total number of atoms with N electrons, where g_j is the degeneracy of configuration j, k is the Boltzmann constant, and T is the temperature.

$$\frac{n_N^j}{n_N} = \frac{g_j e^{-E_j/kT}}{Z_N} \tag{12.21}$$

Z_N denotes the partition function for all ions with N electrons, as defined by:

$$Z_N = \sum_{j=1}^{\infty} g_j \, e^{-(E_j - E_1)/kT} \tag{12.22}$$

The Saha equation below then gives the ratio of successive ion stage densities, where I_N is the ionization energy of the ground state configuration with N electrons. For our project this ionization energy is calculated by taking the difference between successive ground state configuration energy values.

$$\frac{n_{N-1}}{n_N} = \frac{2Z_{N-1}}{n_e Z_N} \left(\frac{2\pi m_e kT}{h^2} \right)^{3/2} e^{-I_N/kT} \tag{12.23}$$

n_e, the number density of free electrons, is used in the Saha equation, but the value of n_e is not an input, it is instead an output set by the charge neutrality condition:

$$n_e = \sum_N (Z - N) n_N \tag{12.24}$$

The mass density is an input for TARTARUS, and therefore, the true densities n_N can be found, instead of just the relative densities from the Saha equation. Because the charge neutrality condition is dependent on n_N and thus the Saha equation, an iterative process must be used to find the n_e and n_N values that satisfy both the charge neutrality condition and the Saha equation. For this project our Saha-Boltzmann post processing script, 'SahaBoltz_calc.py', starts with an initial guess of n_e (unity is used by default), calculates n_N, and then calculates a 'neutral' n_e from the calculated n_N values and the charge neutrality condition. If the original and 'neutral' n_e are within an error factor of each other, (The default value used is 10^{-8}), the ion population is returned. If the two n_e values do not agree, a new initial guess is created from a linear combination of the previous initial guess and 'neutral' n_e, and the process repeats until convergence. (The default linear combination is one-half of each value).

Currently, two inputs need to be manually entered into the post-processing script for it to run correctly. The first is the directory which contains the completed TARTARUS output files, and is assigned to the variable 'direc'. The second is the mass of a single ion, of whatever element is being considered, in units of grams/ion, and this is assigned to the variable 'ion_weight'. The final ionization stage occupation percentages are given in a dictionary 'ion_percent', which maps the ionization stage to occupation percent, and these percentages are plotted against ionization stage by default.

Saha-Boltzmann is ultimately based on classical statistical methods for calculating configuration populations, assuming local thermodynamic equilibrium, and we therefore expect that generally it should preform worse than the variational method, especially in the limit of high density.

12.11 Results

In testing each of the two ionization stage population calculations, we focused on two different materials for the plasma: aluminum and iron. One reference paper we relied on, Faussurier and Blancard (2018), gave results for a solid aluminum plasma, at a temperature of 100 eV, so we started off using the same initial conditions, so that we would have further results to compare to. Another reference paper, Piron and Blenski (2013)), calculated ionization stage populations for iron at varying densities, at a temperature of 40 eV, so we ran some of the same plasma cases with the goal of finding similar results.

For each iron calculation, we considered double excitations up to the 5p subshell, and for each aluminum calculation we considered double excitations up to the 4f subshell. The aluminum excitation level cutoff of 4f follows the cutoff selection in Faussurier and Blancard (2018). For iron, we based our selection of 5p on the TARTARUS average atom calculations.

12.11.1 Iron

In 12.3 we see iron plasma at 40 eV at solid density and at one-tenth solid density. Results from Piron and Blenski (2013) are compared with ChemEOS and our variational method with and without

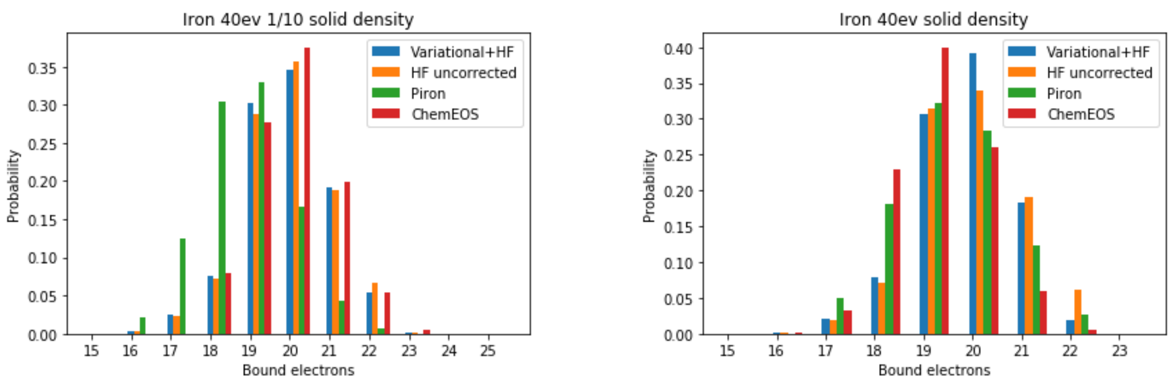


Figure 12.3: Iron plasma at 40 eV at two different densities comparing results from Piron and Blenski (2013), ChemEOS, and our variational method with `ctm_free=.true.` with and without the Hartree-Fock correction

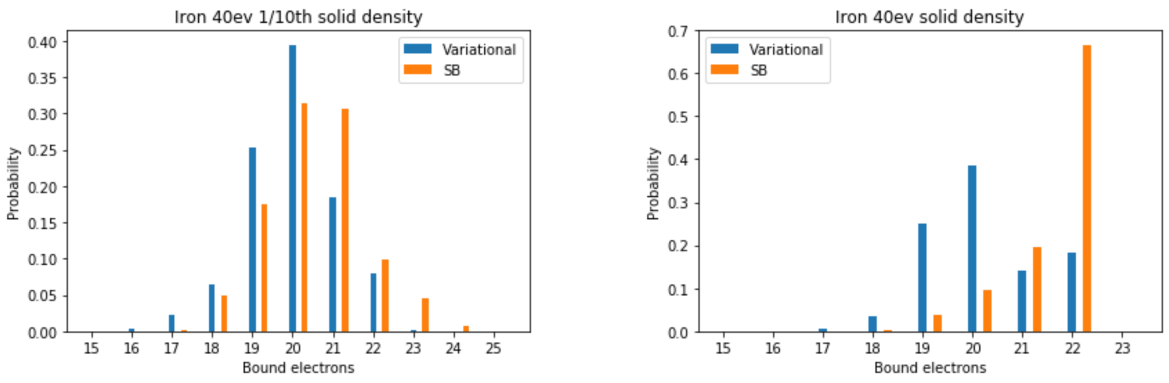
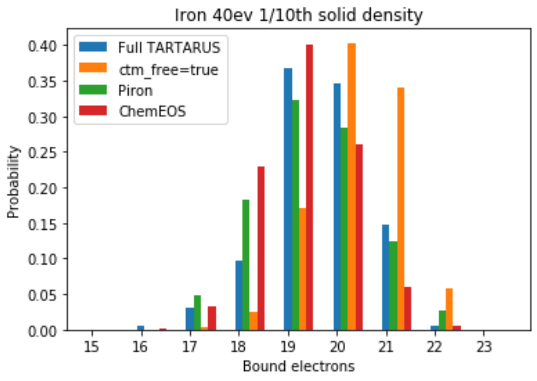


Figure 12.4: Iron plasma at 40 eV at two different densities comparing results from our variational method with Hartree-Fock correction and Saha-Boltzmann

the Hartree-Fock energy correction. At solid density we see that all four methods agree on the general shape of the distribution but still differ substantially. ChemEOS and Piron and Blenski (2013) predict the highest probability at 19 bound electrons, whereas our variational method predicts the highest probability at 20 bound electrons with and without Hartree-Fock. In the one-tenth solid density case we see disagreement between Piron and Blenski (2013) and the other three methods on the peak of the distribution. Unless otherwise specified all TARTARUS results are computed with the ctm_free=true parameter which is a fast approximation that treats unbound electrons as being completely free.

In 12.4 we see the same iron plasmas as in 12.3 with results from our variational method and Saha-Boltzmann. In both cases we see substantial differences in the predicted distributions, in the solid density case especially. We discussed how this may be related to the ideal gas approximation in SB not being valid in this regime, it could also be due to plasma recombination which is not modeled in SB. In 12.5 we see the same iron plasmas again with results comparing the full TARTARUS model and those with `ctm_free=.true.`. At one-tenth solid density we recover the same peak as Piron and Blenski (2013) at 19 bound electrons with the full TARTARUS model where as both



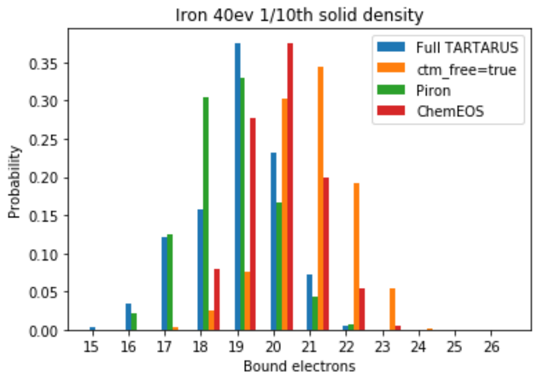


Figure 12.5: Iron plasma at 40 eV at two different densities comparing results from our variational method with `ctm_free=.true.` and `ctm_free=.false.`

ChemEOS and TARTARUS with `ctm_free=.true.` predict the peak at 20 bound electrons. In the solid density case, the full TARTARUS model, ChemEOS and Piron and Blenski (2013) agree on the peak at 19 bound electrons.

12.11.2 Aluminum

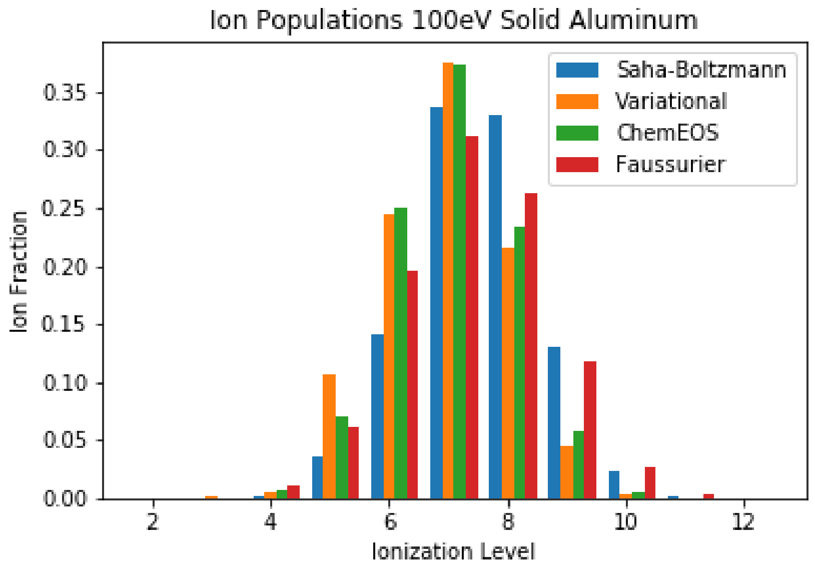
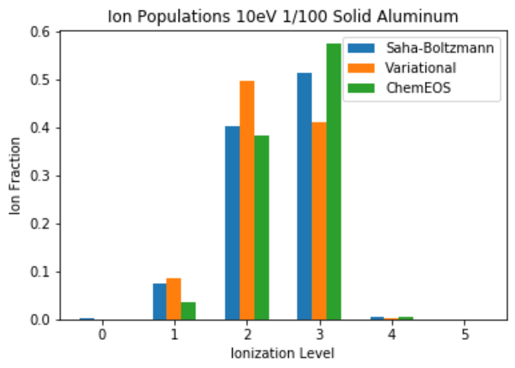


Figure 12.6: Ionization stage occupation populations for 100eV Solid Aluminum, calculated from the Saha-Boltzmann method, the variational method, and from ChemEOS, compared to populations given in Faussurier and Blancard (2018)

We started using solid aluminum at a temperature of 100eV as a test case in order to try and replicate the configuration energy values and ionization stage populations in Faussurier and Blancard (2018).



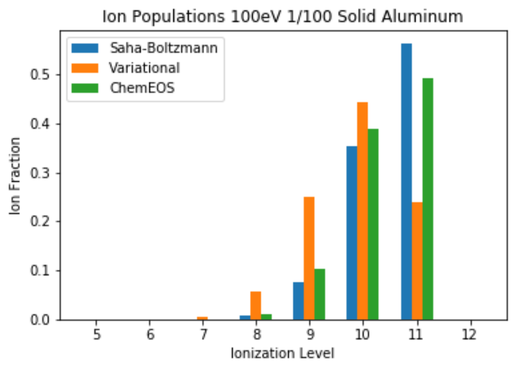


Figure 12.7: Aluminum plasma results at 1/100 of solid density, for temperatures of 10 eV and 100 eV, from the variational method, the Saha-Boltzmann method, and ChemEOS

We were unable to replicate the configuration energy values listed in the paper, however we believe this may be attributed to differing definitions of what quantities contribute to the configuration energy, as we were unable to figure out what Faussurier's precise definition of his E_conf values was.

We did find some success replicating the ionization stage populations given in Faussurier and Blancard (2018). 12.6 shows 100 eV Solid Aluminum, calculated from our two methods, and calculated from ChemEOS, compared to Faussurier's 'Average-Atom Model' results. Generally we found good agreement between all four results. Faussurier additionally provided two partly different ionization stage populations, based on different approximations, however we were unclear what the exact differences were between his three calculations, so we are not certain which population our results should be most similar to.

In addition to solid Aluminum at 100 eV, we tested aluminum at solid density, one-tenth of solid density, and one-hundredth of solid density, at temperatures of 10 eV and 100 eV, in order to see how both the variational and Saha-Boltzmann method preform in varying conditions.

In the case of aluminum at 1/100 solid density, shown in figure 12.7, we find good agreement between our results and the results from ChemEOS. As expected, we see the best agreement between Saha-Boltzmann and ChemEOS in these plots, where density is lower than our other runs.

In the case of aluminum at 1/10 solid density, shown in figure 12.8, we again find overall good agreement, although the variational method appears to have a slightly lower average ionization stage in the case of 100 eV.

In the case of aluminum at solid density, shown in figure 12.9, we start to see a significant difference between the methods. Similar to what is shown in figure 12.3, there is a large discrepancy in the Saha-Boltzmann calculations, most notably when the temperature is 10 eV. Here the variational method and ChemEOS show very strong agreement, while we expect that we are starting to see some of the Saha-Boltzmann assumptions break down, due to the increased density.

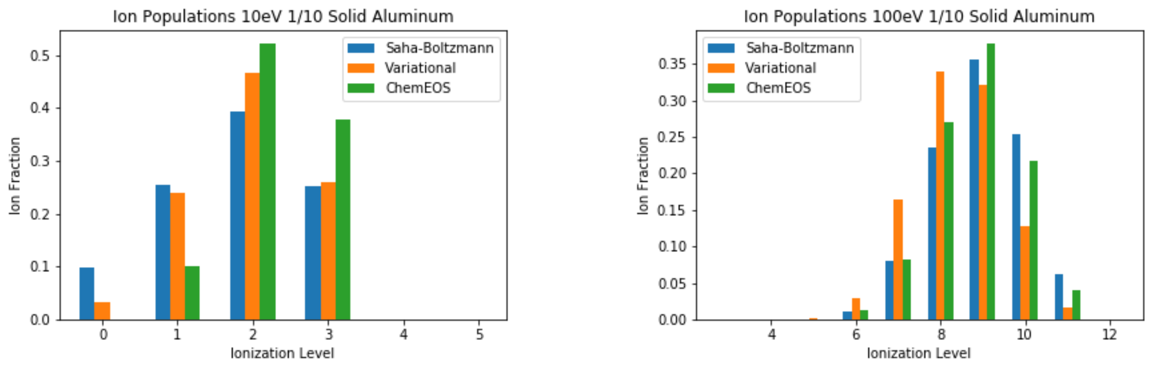


Figure 12.8: Aluminum plasma results at 1/10 of solid density, for temperatures of 10 eV and 100 eV, from the variational method, the Saha-Boltzmann method, and ChemEOS

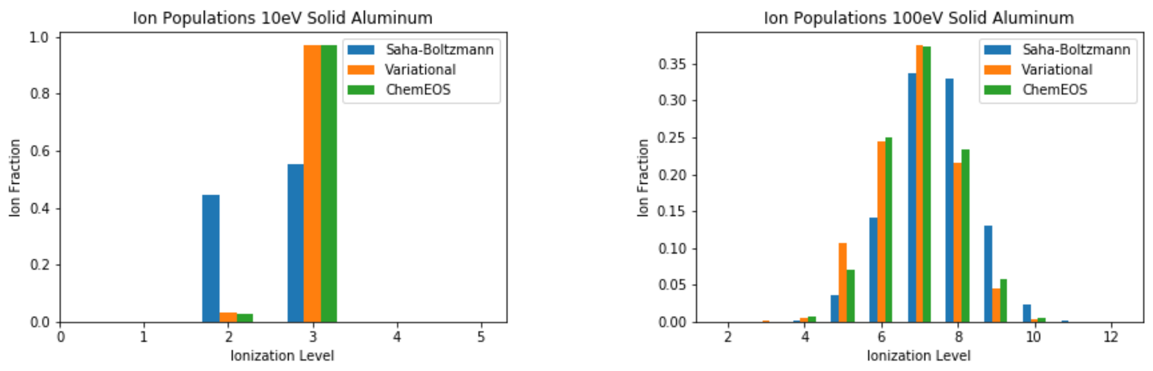


Figure 12.9: Aluminum plasma results at solid density, for temperatures of 10 eV and 100 eV, from the variational method, the Saha-Boltzmann method, and ChemEOS

12.12 Conclusion

After modifying TARTARUS to calculate Hartree-Fock and correlation corrected configuration energies for specific integer occupation configurations, we found promising agreement between our calculated ionization stage populations and those given from both our reference papers and ChemEOS.

We expected that the variational method would calculate more accurate populations than Saha-Boltzmann, given that Saha-Boltzmann is based on classical statistical mechanics. Our results so far appear to support this, as the Saha-Boltzmann method gives significantly different populations in the case of solid density iron plasma at 40 eV, and solid density aluminum plasma at 10 eV.

One possible next step in this work would be to attempt to validate our results with experimental data, ideally with a level of detail where we would be able to distinguish between each of the different methods. Tartarus is currently more computationally expensive than ChemEOS, by a factor of approximately 10-100, so high resolution experimental data would be beneficial to this work, not only to demonstrate broad agreement, but also to identify regimes where TARTARUS calculates significantly more accurate populations, compared to ChemEOS.

Another consideration for future work is that our results are sensitive to the value of the free electron density used. Most of our runs shown here are run without continuum electron, and only bound and free electrons, however if continuum electrons are considered, as with the 'Full' TARTARUS method discussed earlier, then the populations will be dependent on whether or not the continuum electrons are considered a part of free electron density.

Another avenue to continue this work would be to use the configuration populations calculated using our two methods to calculate opacities. The overarching goal for this project was to ultimately be able to use the TARTARUS code to calculate more accurate opacities, and comparing calculated opacity values from TARTARUS to observational or predicted opacity values could provide additional validation for the accuracy of the TARTARUS methods.

Finally, one more promising avenue for future work might be to explore how we can speed up our calculations. In this project we did not focus significantly on how we could most efficiently calculate the energy corrections, and in each plasma we ran TARTARUS for many configurations which did not contribute significantly to the final populations. There may be potential optimizations in our implemented calculations and electron configuration selection.

Jackson White *is an rising senior at Rice University, majoring in Astrophysics. His current research work includes the creation and collision of magnetized plasma jets, and the Carina Nebula. He plans to pursue a PhD in Astrophysics at the conclusion of his undergraduate career.*

William Johns *received his B.S. in Physics and M.S. in Mathematics from Montana State University in 2010 and 2013 respectively. He is currently a PhD candidate in Mathematics at Montana State University studying rational approximations for use in frequency dependent line modeling. He hopes to work for a National Lab or in private industry on clean/renewable energy applications*

Bibliography

Robert D. Cowan. *The theory of atomic structure and spectra*. 1981.

Gérald Faussurier and Christophe Blancard. Density effects on electronic configurations in dense plasmas. *Phys. Rev. E*, 97:023206, Feb 2018. doi: 10.1103/PhysRevE.97.023206. URL `https://link.aps.org/doi/10.1103/PhysRevE.97.023206`.

Peter Hakel and David P. Kilcrease. Chemeos: A new chemical-picture-based model for plasma equation-of-state calculations. *AIP Conference Proceedings*, 730(1):190–199, 2004. doi: 10.1063/1.1824870. URL `https://aip.scitation.org/doi/abs/10.1063/1.1824870`.

R. Piron and T. Blenski. Variational average-atom in quantum plasmas (vaaqp) – application to radiative properties. *High Energy Density Physics*, 9(4):702 – 710, 2013. ISSN 1574-1818. doi: https://doi.org/10.1016/j.hedp.2013.07.002. URL `http://www.sciencedirect.com/science/article/pii/S1574181813001626`.

N.R. Shaffer, N.G. Ferris, J. Colgan, D.P. Kilcrease, and C.E. Starrett. Free-free opacity in dense plasmas with an average atom model. *High Energy Density Physics*, 23:31 – 37, 2017. ISSN 1574-1818. doi: https://doi.org/10.1016/j.hedp.2017.02.008. URL `http://www.sciencedirect.com/science/article/pii/S1574181817300137`.

C.E. Starrett. A green's function quantum average atom model. *High Energy Density Physics*, 16: 18 – 22, 2015. ISSN 1574-1818. doi: https://doi.org/10.1016/j.hedp.2015.05.001. URL `http://www.sciencedirect.com/science/article/pii/S1574181815000439`.

Brian G. Wilson, David A. Liberman, and Paul T. Springer. A deficiency of local density functionals for the calculation of self-consistent field atomic data in plasmas. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 54(5):857 – 878, 1995. ISSN 0022-4073. doi: https://doi.org/10.1016/0022-4073(95)00104-S. URL `http://www.sciencedirect.com/science/article/pii/002240739500104S`.